C.E.C.I

Consortium des Equipements
de Calcul Intensif
en Fédération Wallonie-Bruxelles

Preparing, submitting and managing jobs with Slurm

damien.francois@uclouvain.be
October 2020

UCL Université catholique de Louvain

INSTITUT DE CALCUL INTENSIF ET DE STOCKAGE DE MASSE

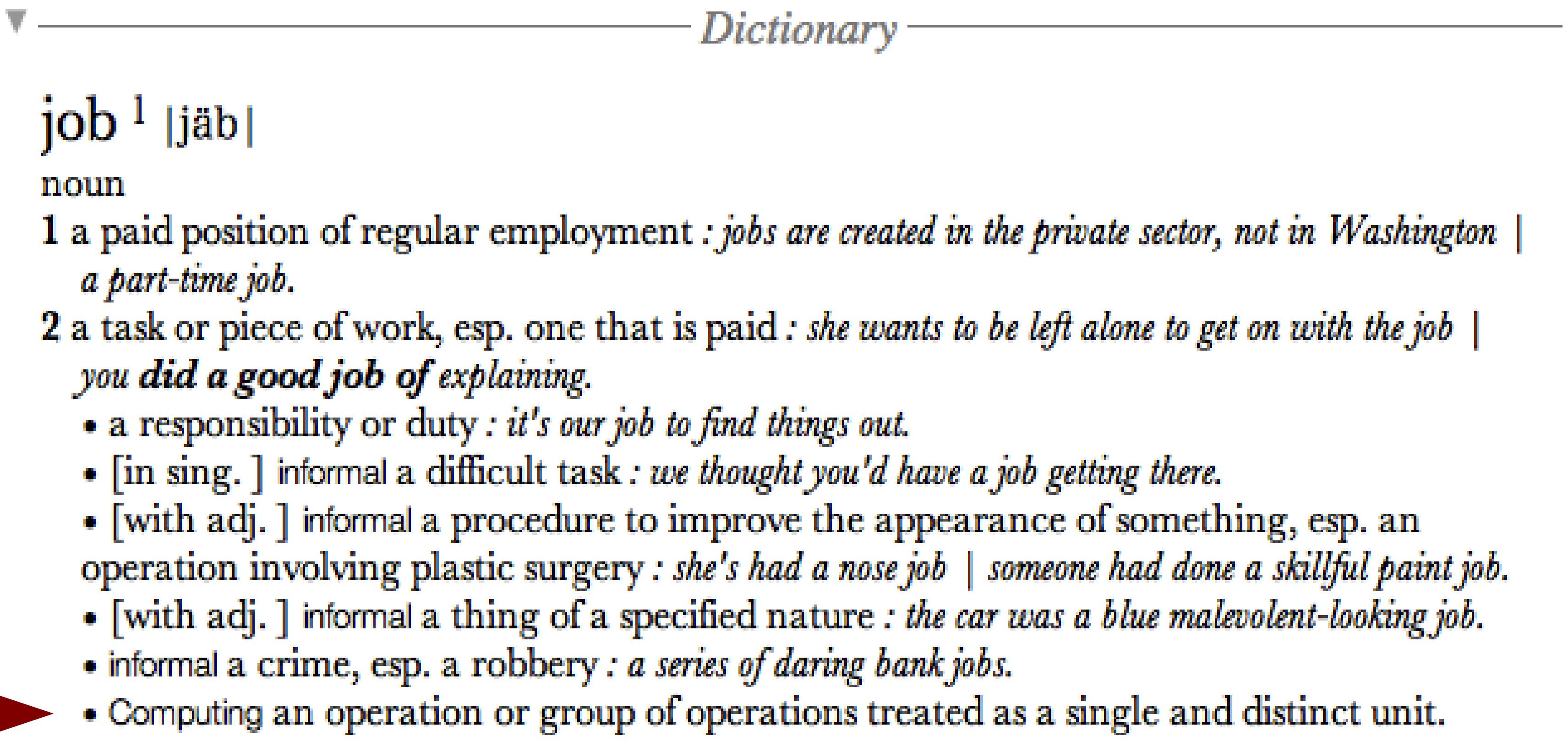
Until now:

- access the cluster ✓
- copy data to/from the cluster ✓
- create parallel software ✓
- compile code and use optimized libraries ✓
- actually run software on the cluster ?

tl;dr:

- submit a *job* to the *scheduler*

What is a job?



Dictionary

job [1] |jäb|

noun

1 a paid position of regular employment : *jobs are created in the private sector, not in Washington* | *a part-time job.*

2 a task or piece of work, esp. one that is paid : *she wants to be left alone to get on with the job* | *you* ***did a good job of*** *explaining.*

- a responsibility or duty : *it's our job to find things out.*
- [in sing.] informal a difficult task : *we thought you'd have a job getting there.*
- [with adj.] informal a procedure to improve the appearance of something, esp. an operation involving plastic surgery : *she's had a nose job* | *someone had done a skillful paint job.*
- [with adj.] informal a thing of a specified nature : *the car was a blue malevolent-looking job.*
- informal a crime, esp. a robbery : *a series of daring bank jobs.*
- Computing an operation or group of operations treated as a single and distinct unit.

Job scheduler/Resource manager :

Piece of software which:

- manages and allocates resources;
- manages and schedules jobs;

Two computers are available for 10h

Your job runs now, then yours. You wait.

- and sets up the environment for parallel and distributed computing.

Resources:

CPU cores

Memory

Disk space

Network

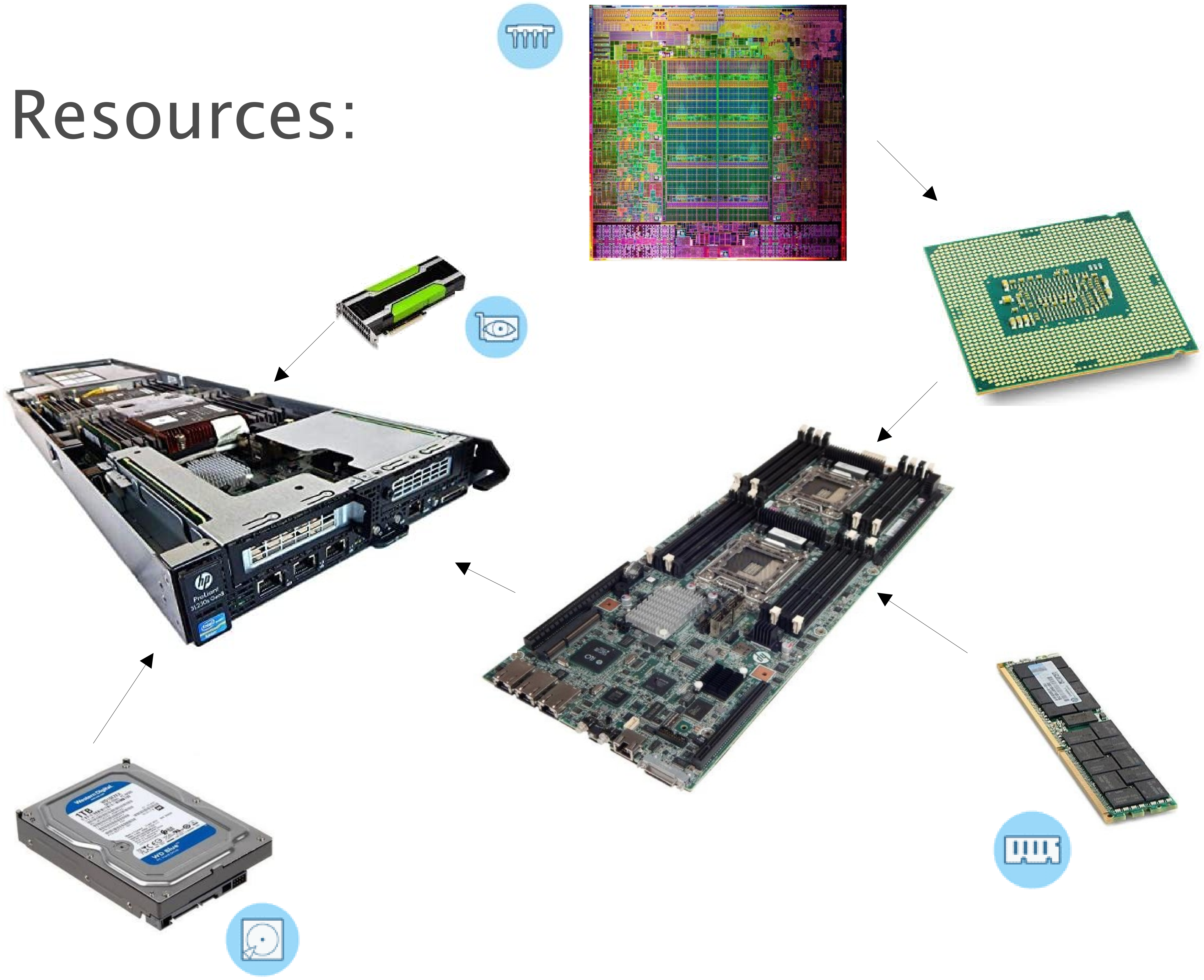
Accelerators

Software

Licenses

Resources:

Resources:



slurm

workload manager

Slurm

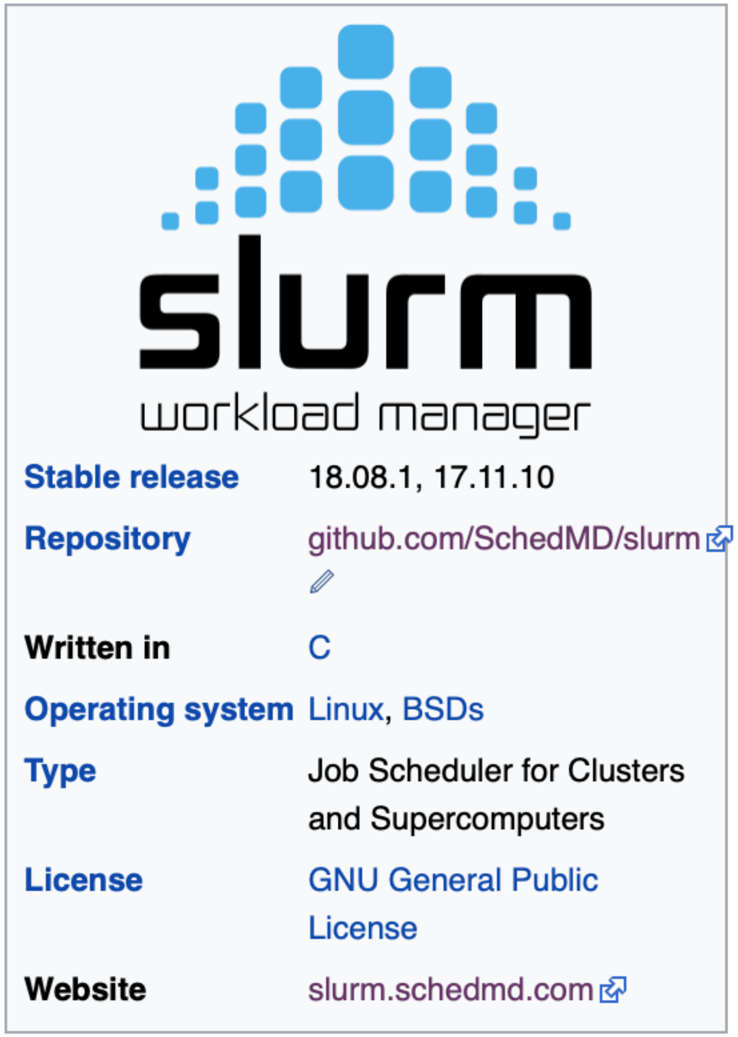
Free and open-source

Mature (exists since ~2003)

Very active community

Many success stories

Widely used



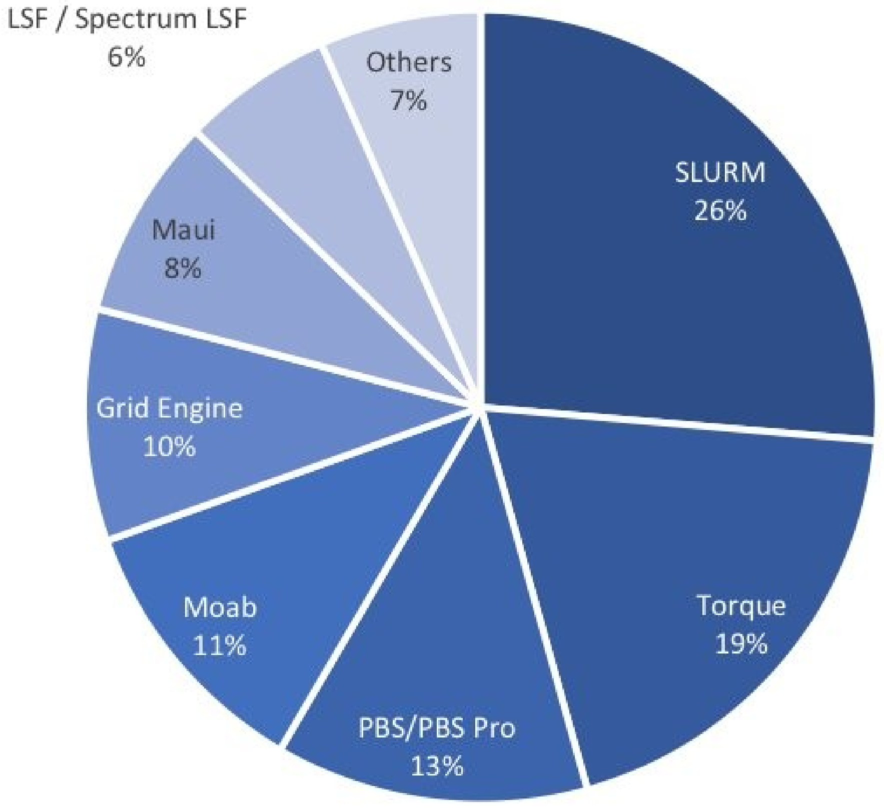
Stable release	18.08.1, 17.11.10
Repository	github.com/SchedMD/slurm
Written in	C
Operating system	Linux, BSDs
Type	Job Scheduler for Clusters and Supercomputers
License	GNU General Public License
Website	slurm.schedmd.com

Also an intergalactic soft drink

Futurama (TV Series, creators David X. Cohen, Matt Groening)
Fry and the Slurm Factory (1999)
20th Century Fox Television

Reported Job Management Packages at HPC Sites

Intersect360 Research, *HPC User Site Census: Middleware and Developer Tools*, 2019



You will learn how to:

Create a job
Monitor the jobs
Control your own job
Get job accounting info

with

slurm
workload manager

1. Make up your mind

e.g. 1 core, 2GB RAM for 1 hour

Job parameters

- resources you need;
- operations you need to perform.

e.g. launch 'myprog'

Job steps

2. Write a submission script

It is a shell script (Bash)

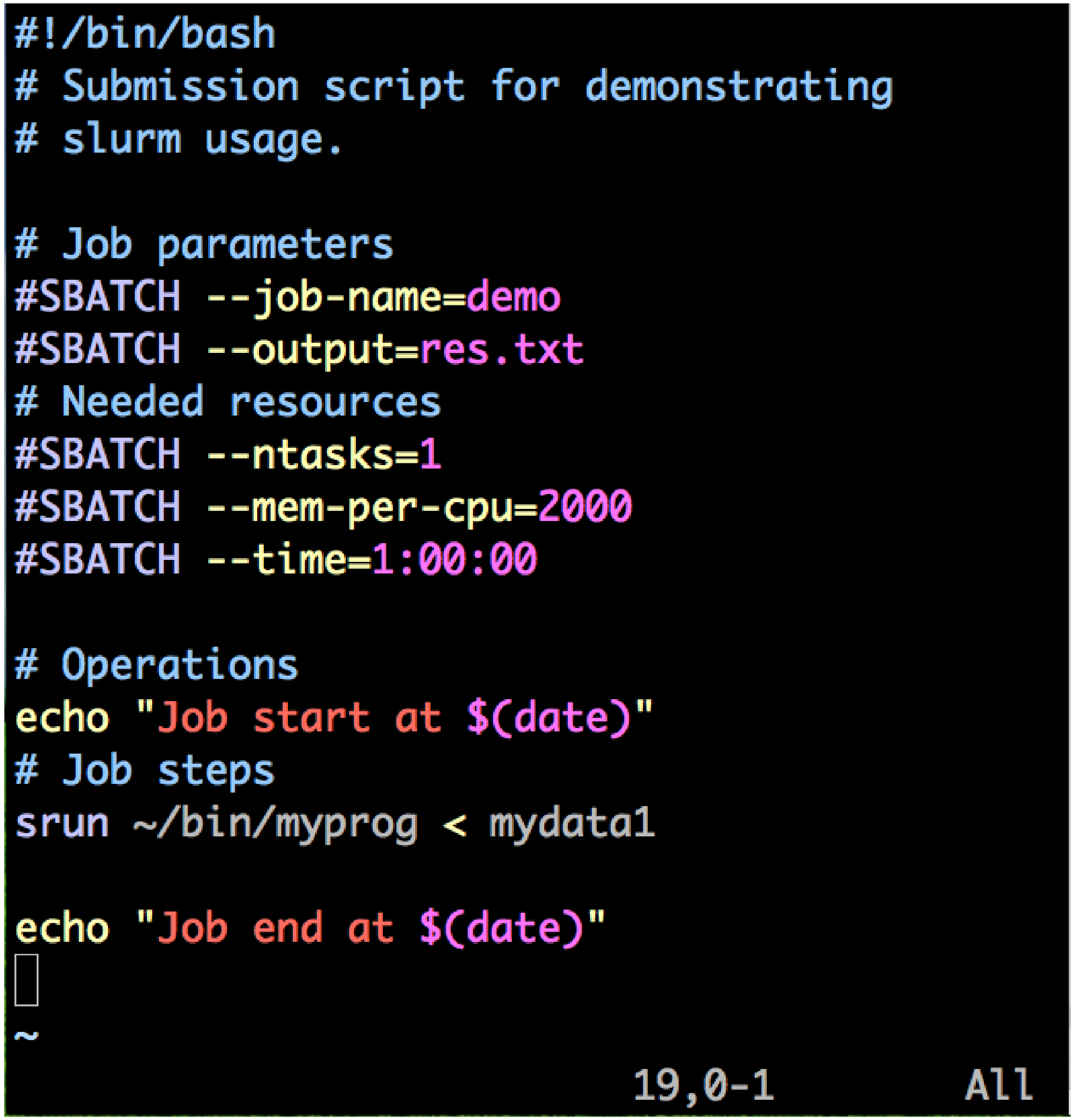
Bash sees these as comments

Slurm takes them as parameters

Job step creation

Regular Bash comment

Regular Bash commands



```
#!/bin/bash
# Submission script for demonstrating
# slurm usage.

# Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
# Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00

# Operations
echo "Job start at $(date)"
# Job steps
srun ~/bin/myprog < mydata1

echo "Job end at $(date)"

~
                                  19,0-1        All
```

2. Write a submission script

It is a shell script (Bash)

Bash sees these as comments

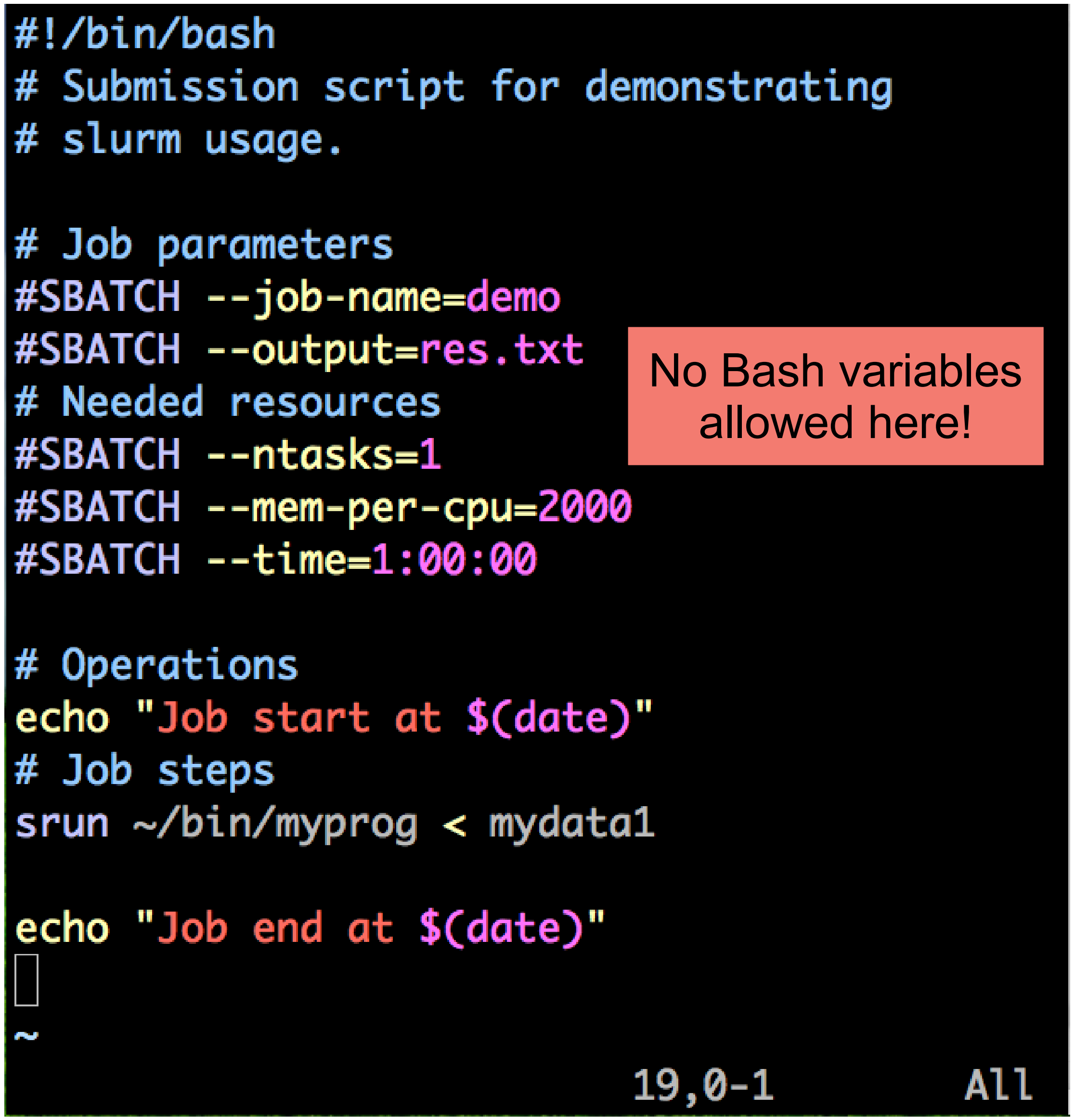
Slurm takes them as parameters

Job step creation

Regular Bash comment

Regular Bash commands

No Bash variables allowed here!



```
#!/bin/bash
# Submission script for demonstrating
# slurm usage.

# Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
# Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00

# Operations
echo "Job start at $(date)"
# Job steps
srun ~/bin/myprog < mydata1

echo "Job end at $(date)"

~
                         19,0-1        All
```

Constraints and resources

You want	You ask
To choose a specific feature (e.g. a processor type or a network type)	--constraint
To use a specific resources (e.g. a GPU)	--gres
To access a specific licensed software	--licence
To chose a partition	--partition
To use a specific QOS	--qos
To choose the CPU distribution on nodes	--nodes --ntasks-per-nodes --cpus-per-tasks

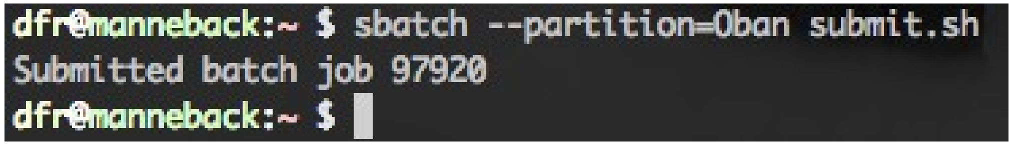
Other useful parameters

You want	You ask
To set a job name	--job-name=MyJobName
To attach a comment to the job	--comment=”Some comment”
To get emails	--mail-type= BEGIN|END|FAILED|ALL|TIME_LIMIT_90 --mail-user=my@mail.com
To set the name of the ouptut file	--output=result-%j.txt --error=error-%j.txt
To get an idea of when it would start	--test-only
To specify an ordering of your jobs	--dependency=after(ok|notok|any):jobids --dependency=singleton

3. Submit the script

I submit with
'sbatch'

One more
job parameter



```
dfr@manneback:~ $ sbatch --partition=Oban submit.sh
Submitted batch job 97920
dfr@manneback:~ $
```

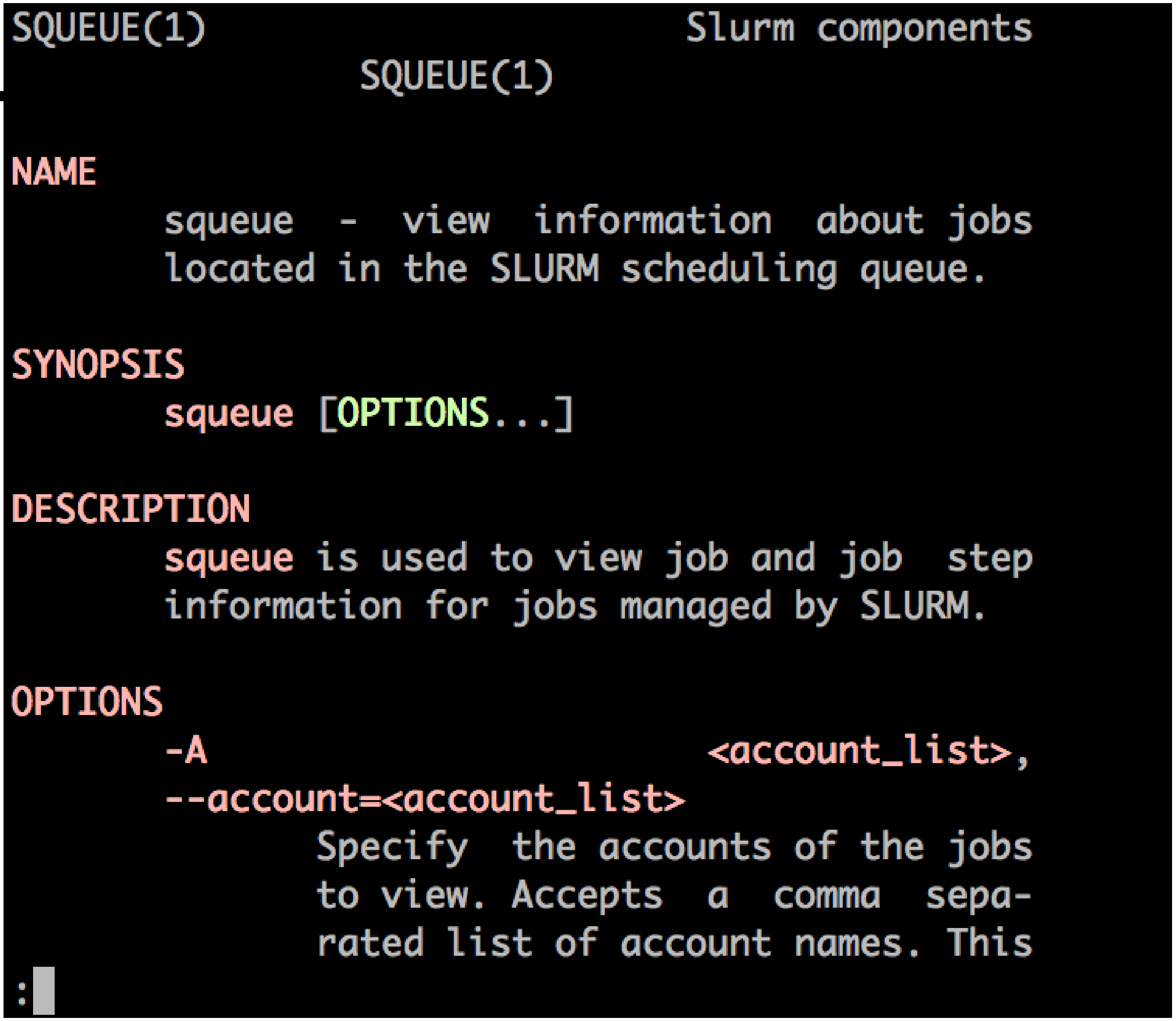
Slurm gives
me the JobID

Submit your first job!

1. Connect to a cluster
2. Open a text editor and write the script for a job that will run the "hostname" command
3. Submit the job
4. Look for files created in your directory

4. Monitor your job

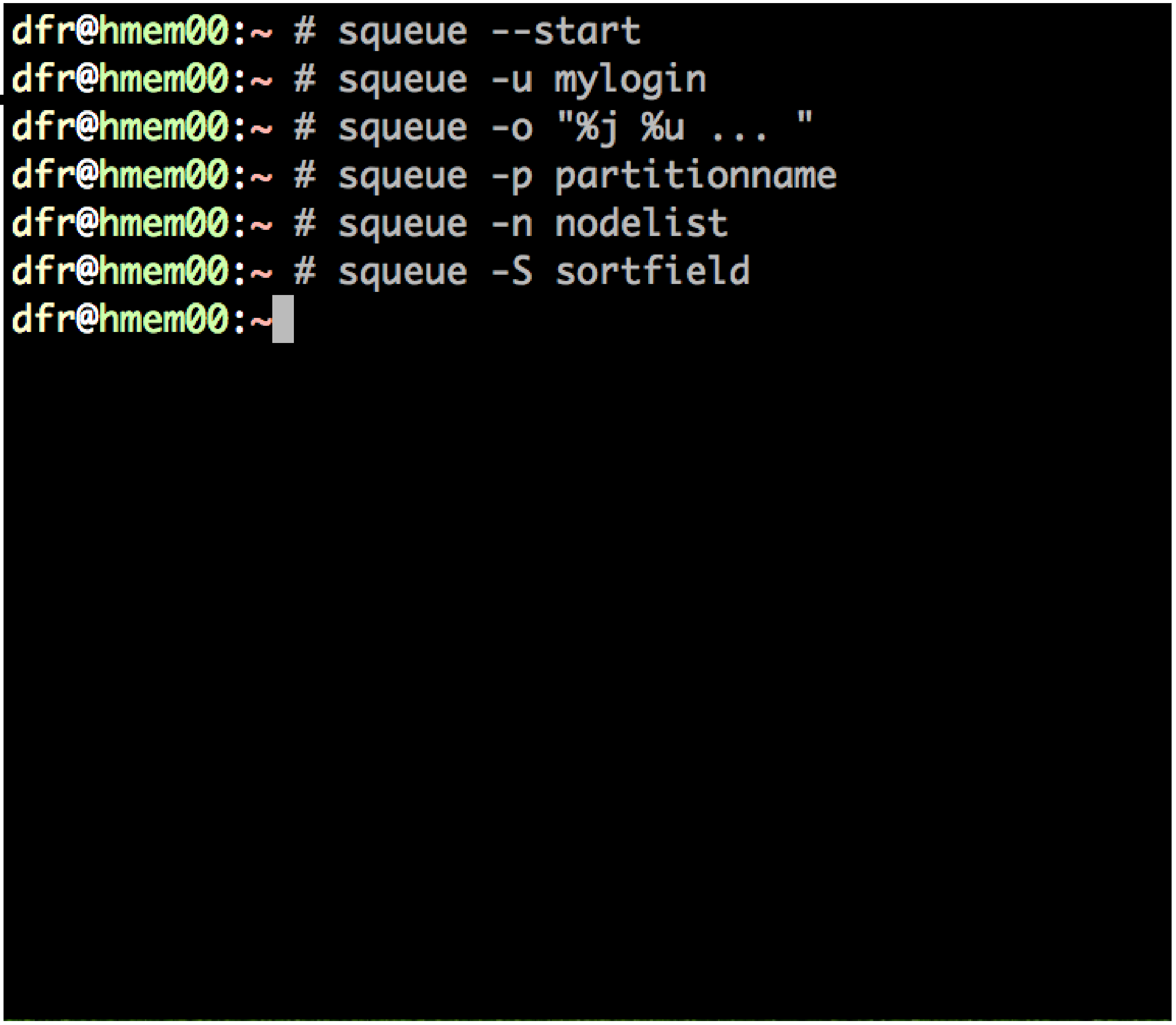
- squeue
- sprio
- sstat
- sview



```
SQUEUE(1)                    Slurm components
                SQUEUE(1)

NAME
       squeue  -  view  information  about jobs
       located in the SLURM scheduling queue.

SYNOPSIS
       squeue [OPTIONS...]

DESCRIPTION
       squeue is used to view job and job  step
       information for jobs managed by SLURM.

OPTIONS
       -A                        <account_list>,
       --account=<account_list>
              Specify  the accounts of the jobs
              to view. Accepts  a  comma  sepa-
              rated list of account names. This
:
```

4. Monitor your job

- squeue
- sprio
- sstat
- sview



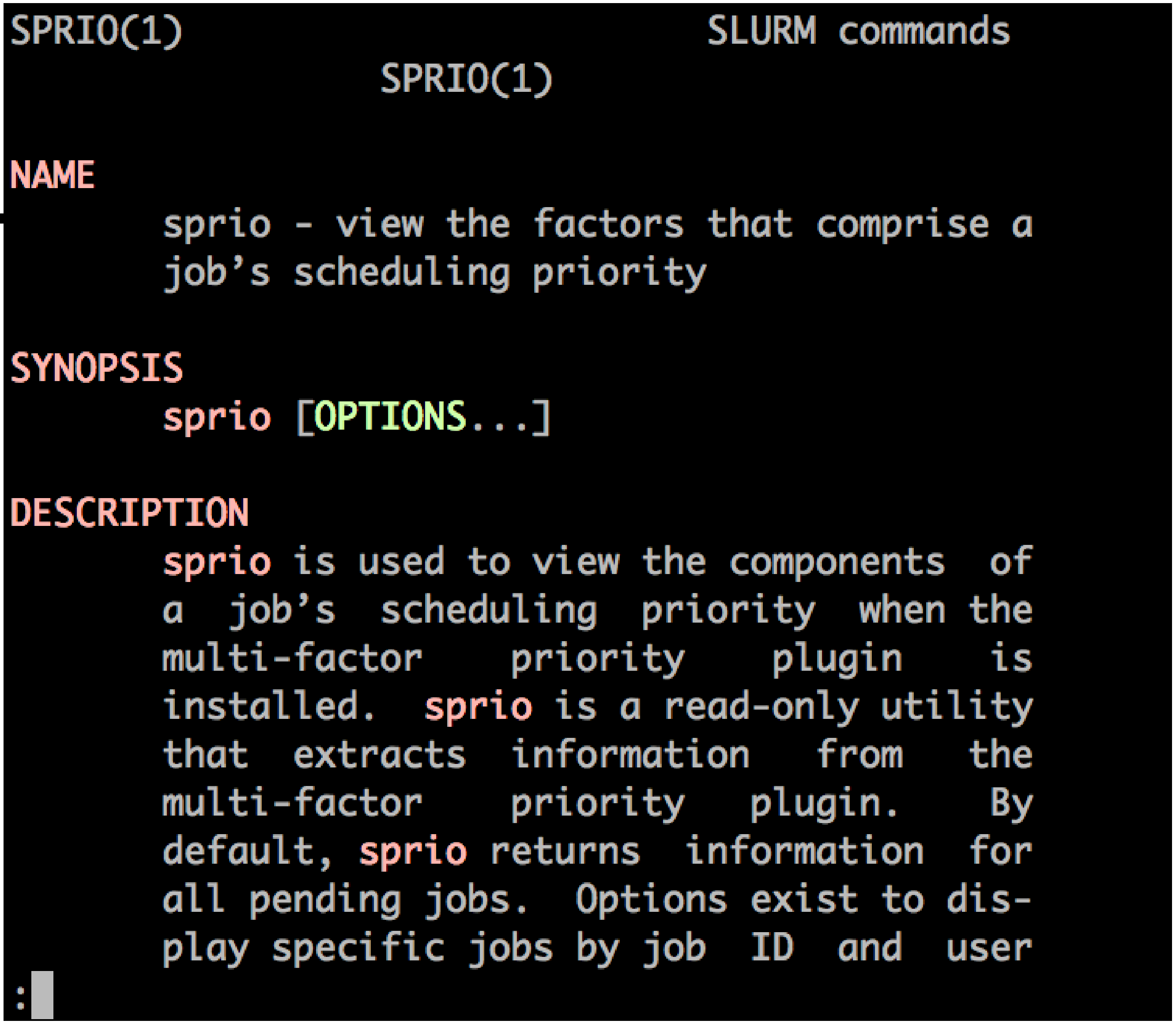
```
dfr@hmem00:~ # squeue --start
dfr@hmem00:~ # squeue -u mylogin
dfr@hmem00:~ # squeue -o "%j %u ... "
dfr@hmem00:~ # squeue -p partitionname
dfr@hmem00:~ # squeue -n nodelist
dfr@hmem00:~ # squeue -S sortfield
dfr@hmem00:~
```

Submit your second job!

1. Connect to a cluster
2. Open a text editor and write the script for a job that will run the “sleep 3000” command and request a 5 minutes run time .
3. Submit the job (on a debug partition)
4. Look for files created in your directory

4. Monitor your job

- squeue
- sprio
- sstat
- sview

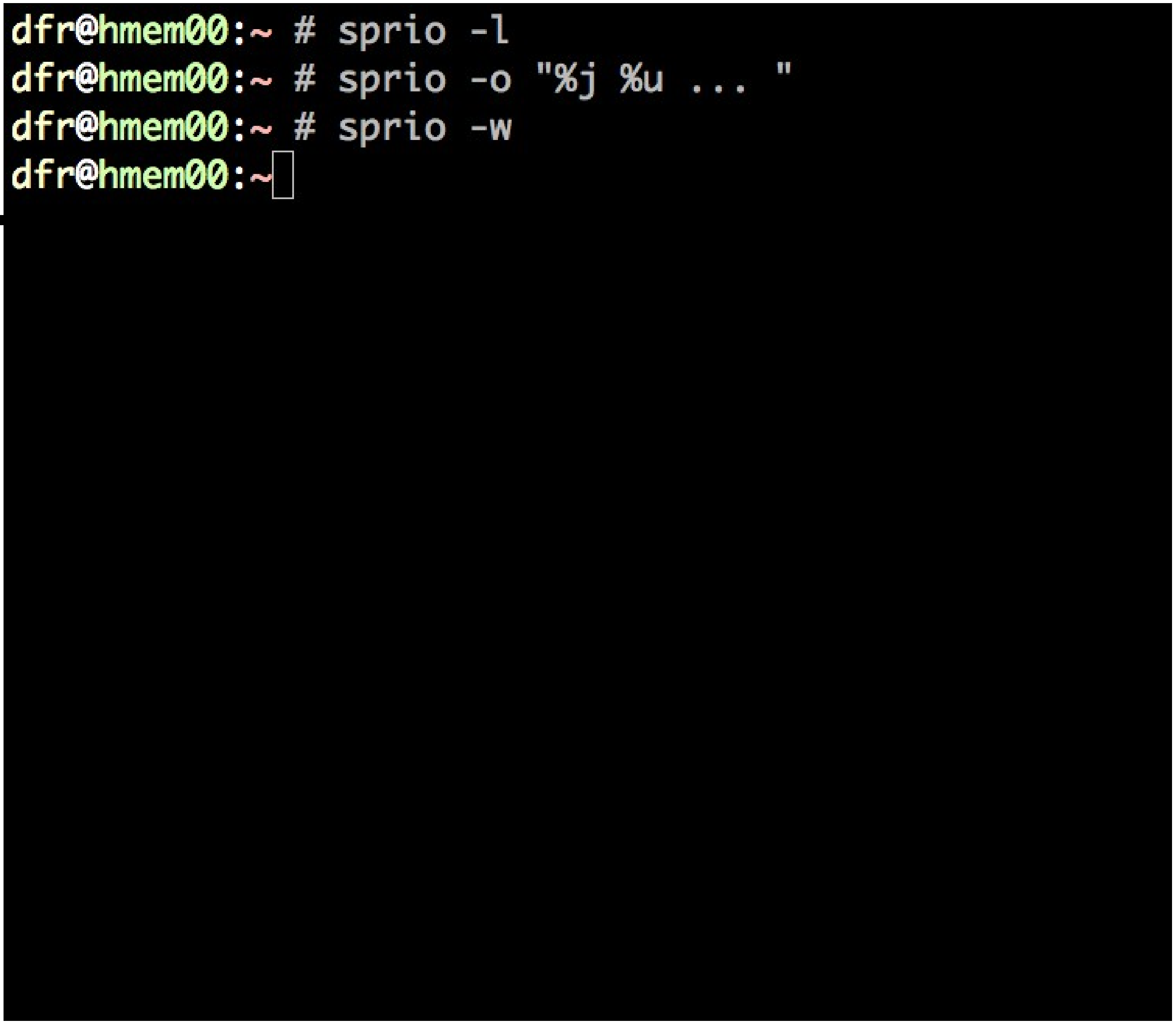


```
SPRIO(1)                  SLURM commands
              SPRIO(1)

NAME
       sprio - view the factors that comprise a
       job's scheduling priority

SYNOPSIS
       sprio [OPTIONS...]

DESCRIPTION
       sprio is used to view the components  of
       a  job's  scheduling  priority  when the
       multi-factor    priority    plugin    is
       installed.  sprio is a read-only utility
       that  extracts  information   from   the
       multi-factor    priority   plugin.    By
       default, sprio returns  information  for
       all pending jobs.  Options exist to dis-
       play specific jobs by job  ID  and  user
:
```

4. Monitor your job

- squeue
- sprio
- sstat

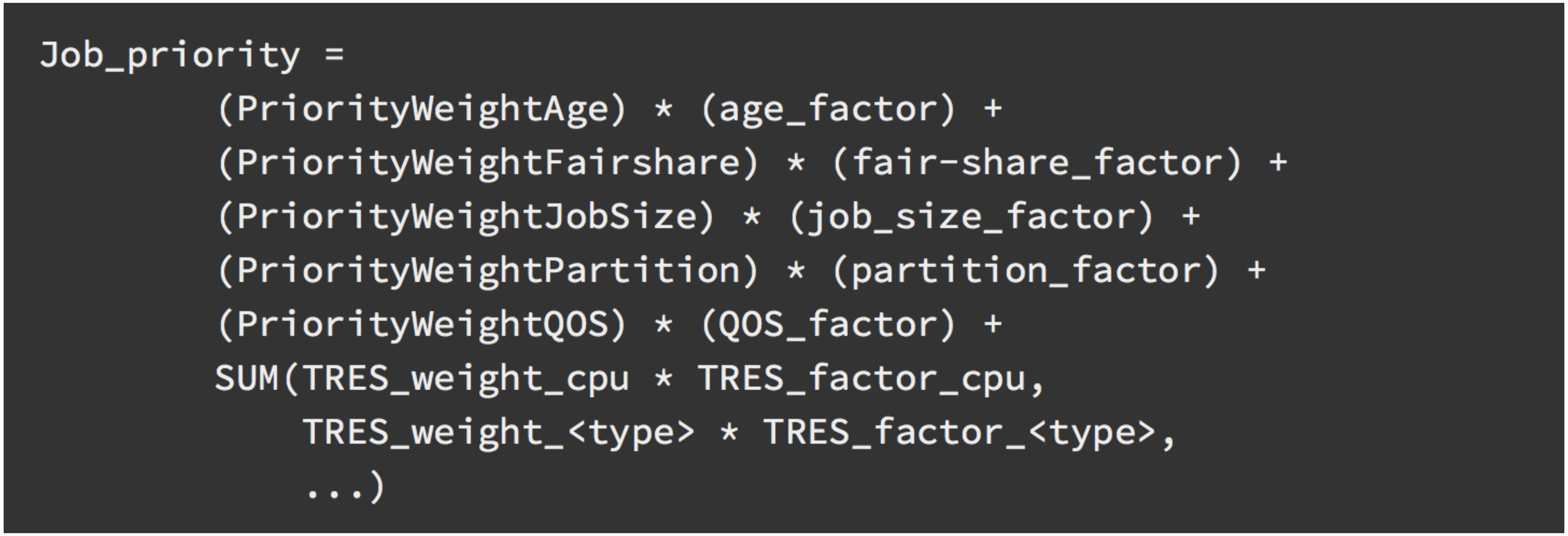
- sview



```
dfr@hmem00:~ # sprio -l
dfr@hmem00:~ # sprio -o "%j %u ... "
dfr@hmem00:~ # sprio -w
dfr@hmem00:~
```

A word about priority

Slurm reserves resources for the top priority job of each partition



```
Job_priority =
        (PriorityWeightAge) * (age_factor) +
        (PriorityWeightFairshare) * (fair-share_factor) +
        (PriorityWeightJobSize) * (job_size_factor) +
        (PriorityWeightPartition) * (partition_factor) +
        (PriorityWeightQOS) * (QOS_factor) +
        SUM(TRES_weight_cpu * TRES_factor_cpu,
            TRES_weight_<type> * TRES_factor_<type>,
            ...)
```

```
dfr@hmem00:~ $ sprio -w
          JOBID   PRIORITY         AGE  FAIRSHARE
        Weights                500000000 1000000000
```

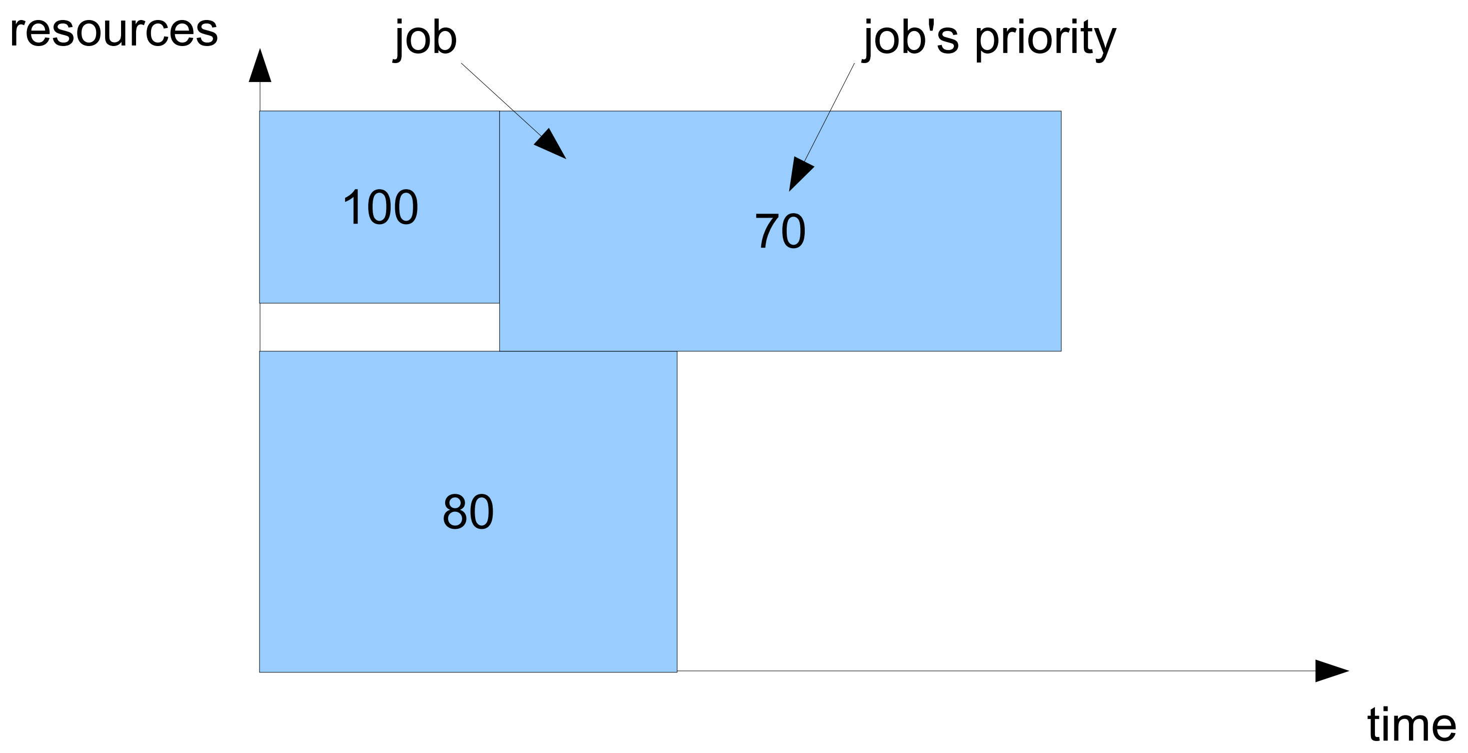
https://slurm.schedmd.com/priority_multifactor.html

Check the priority settings

1. Connect to a cluster
2. Run “sprio –w”
3. Run “scontrol show config | grep ^Priority”
4. Look for the meaning of the items with “man slurm.conf” (Searching is done with “/”)

A word about backfill

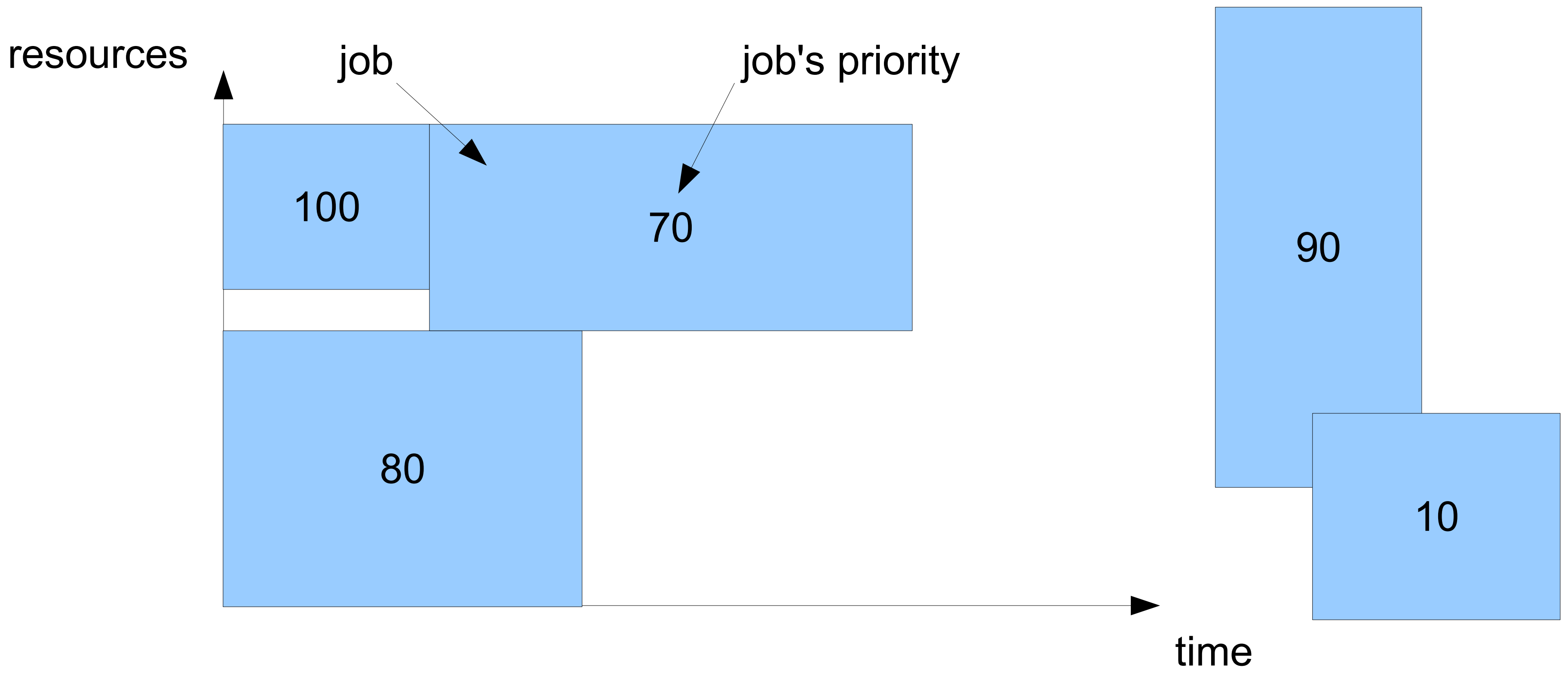
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



A job is a number of cpus times duration

A word about backfill

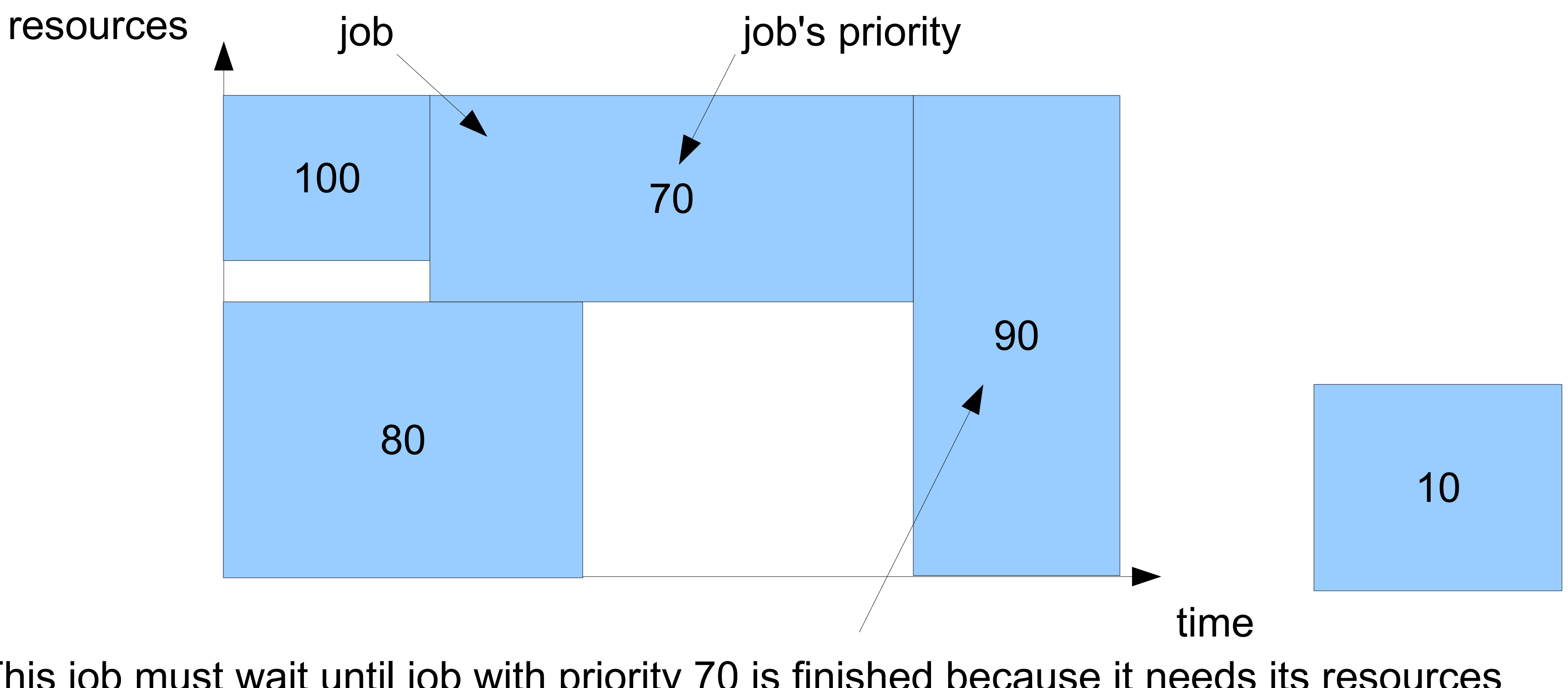
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



Two more jobs to schedule

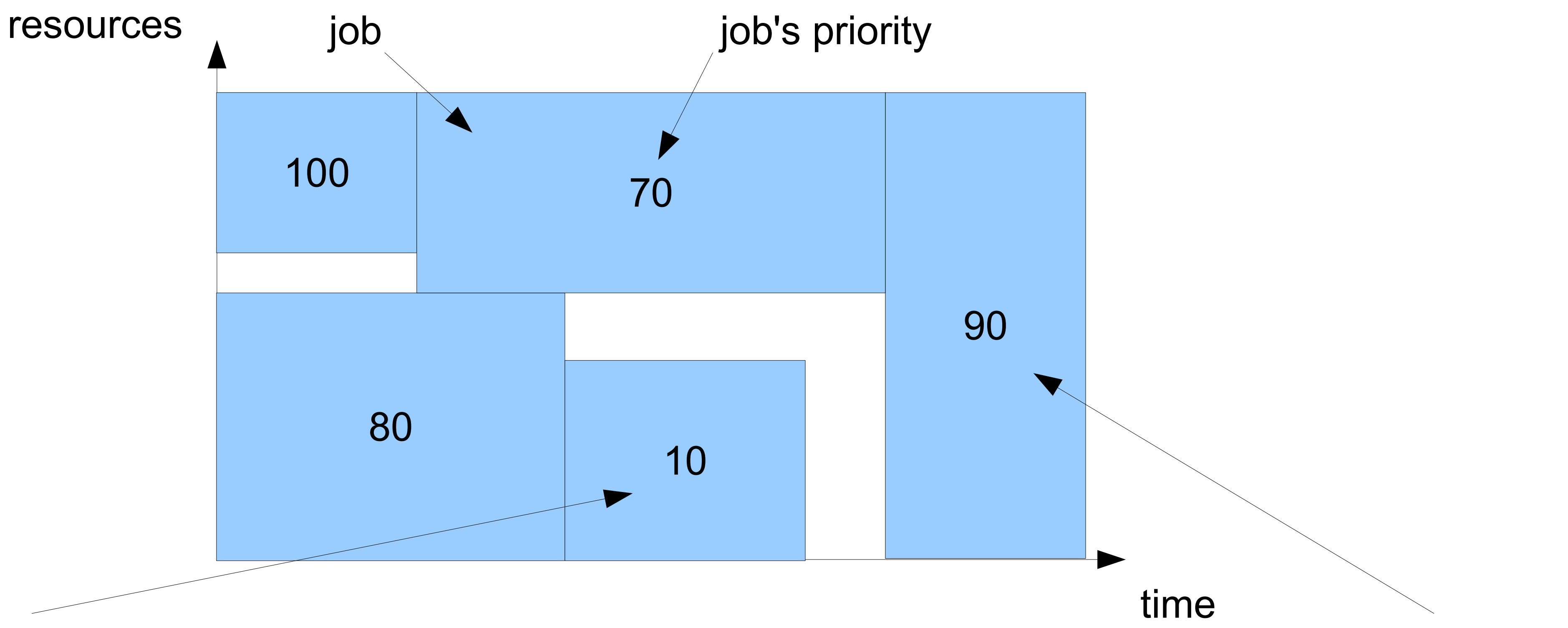
A word about backfill

The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.


resources
job
job's priority
100
70
90
80
10
time
This job must wait until job with priority 70 is finished because it needs its resources

A word about backfill

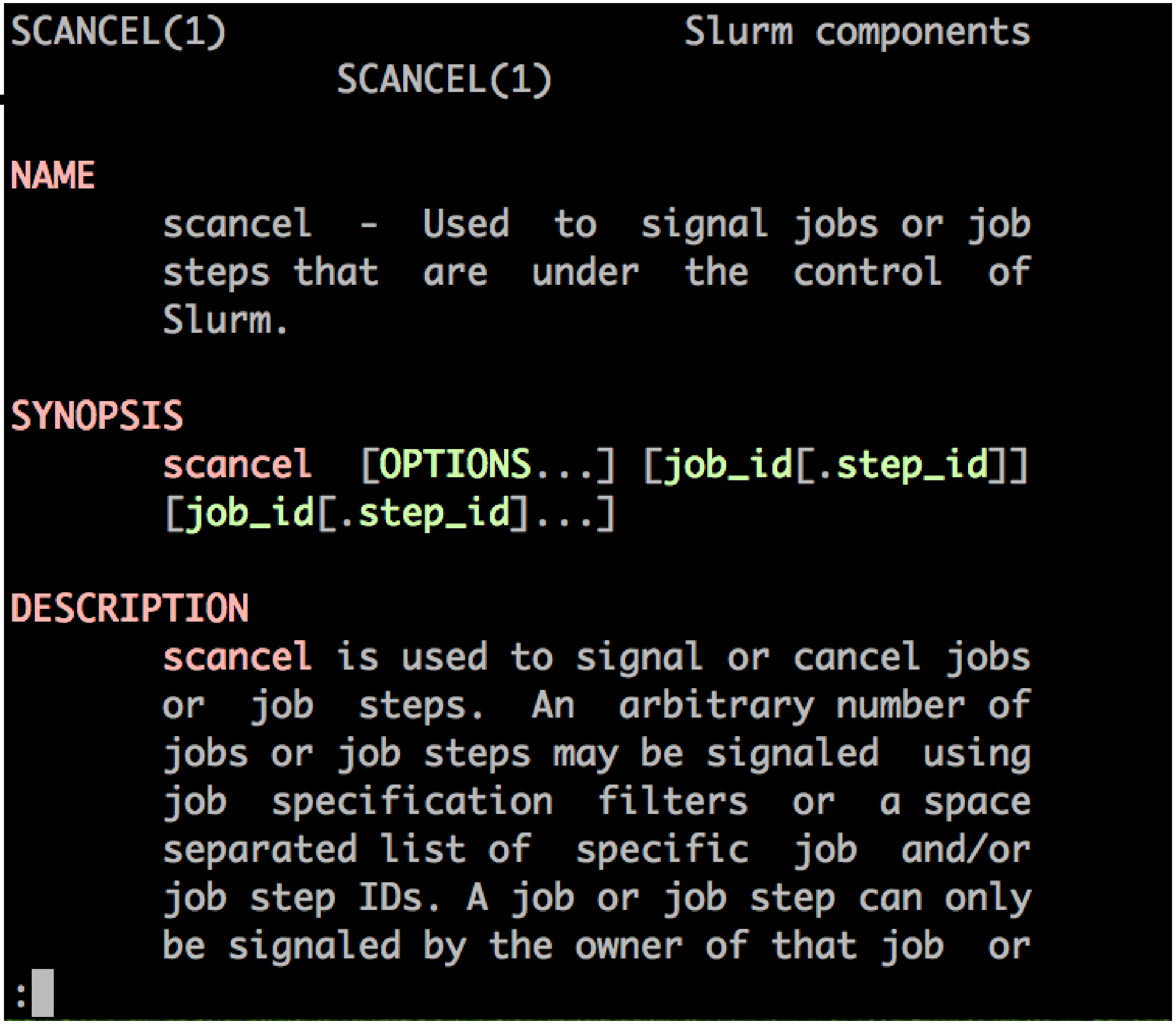
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



Low priority job has short max run time and less requirements ; it starts before larger priority job

5. Control your job

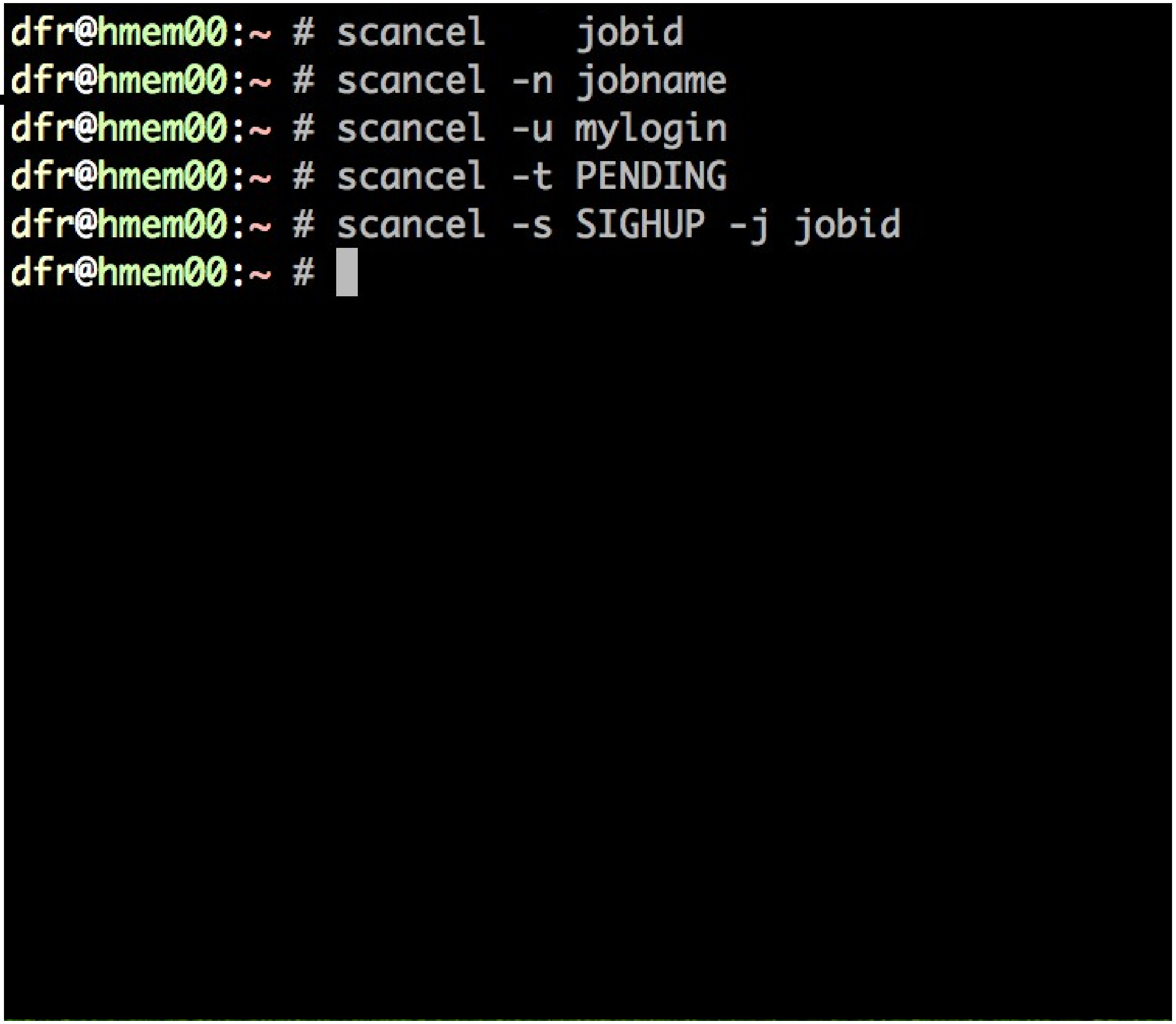
- scancel
- scontrol
- sview



```
SCANCEL(1)                      Slurm components
              SCANCEL(1)

NAME
       scancel  -  Used  to  signal jobs or job
       steps that  are  under  the  control  of
       Slurm.

SYNOPSIS
       scancel  [OPTIONS...] [job_id[.step_id]]
       [job_id[.step_id]...]

DESCRIPTION
       scancel is used to signal or cancel jobs
       or  job  steps.  An  arbitrary number of
       jobs or job steps may be signaled  using
       job  specification  filters  or  a space
       separated list of  specific  job  and/or
       job step IDs. A job or job step can only
       be signaled by the owner of that job  or
:
```

5. Control your job

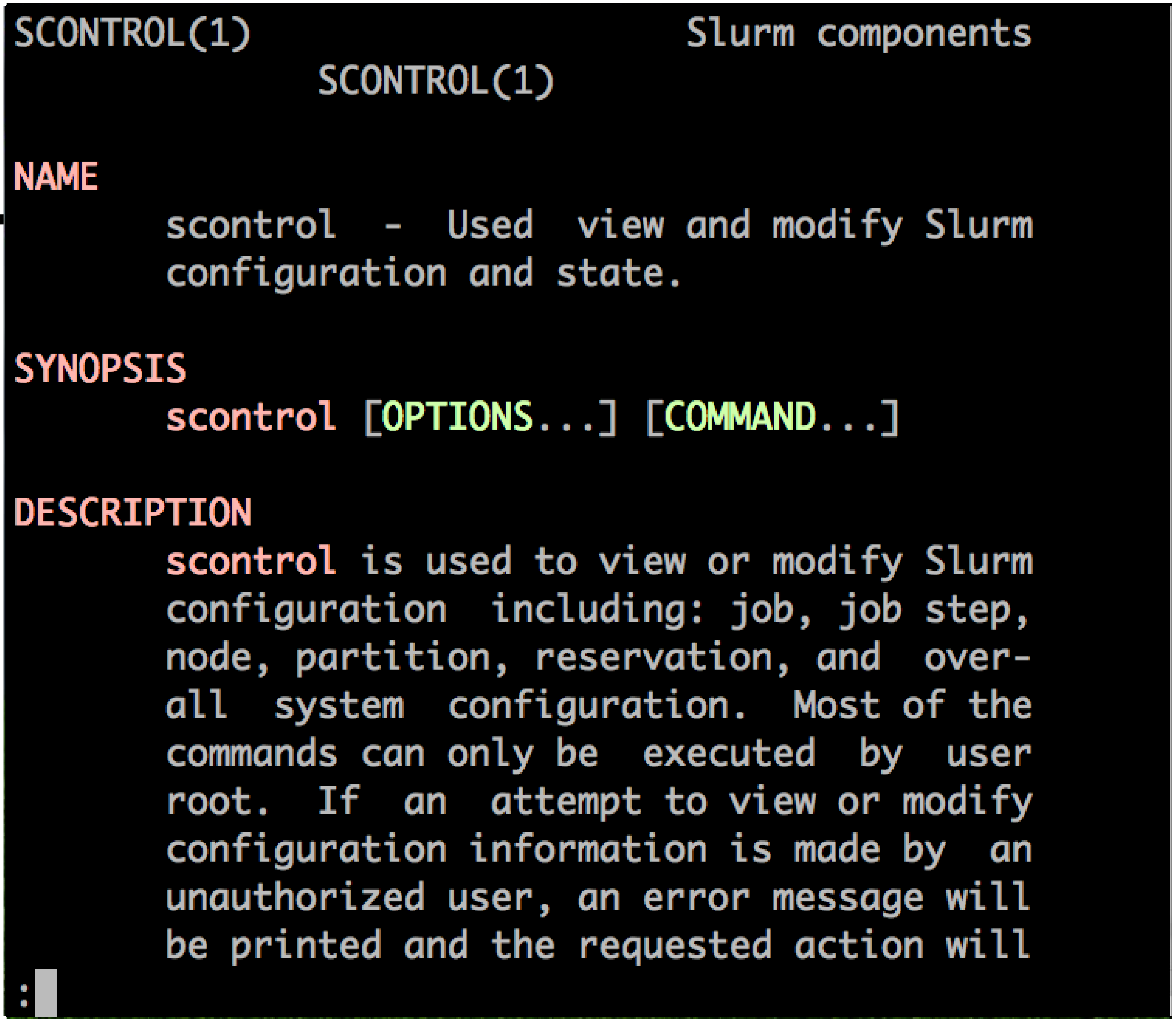
- scancel
- scontrol
- sview



```
dfr@hmem00:~ # scancel    jobid
dfr@hmem00:~ # scancel -n jobname
dfr@hmem00:~ # scancel -u mylogin
dfr@hmem00:~ # scancel -t PENDING
dfr@hmem00:~ # scancel -s SIGHUP -j jobid
dfr@hmem00:~ #
```

5. Control your job

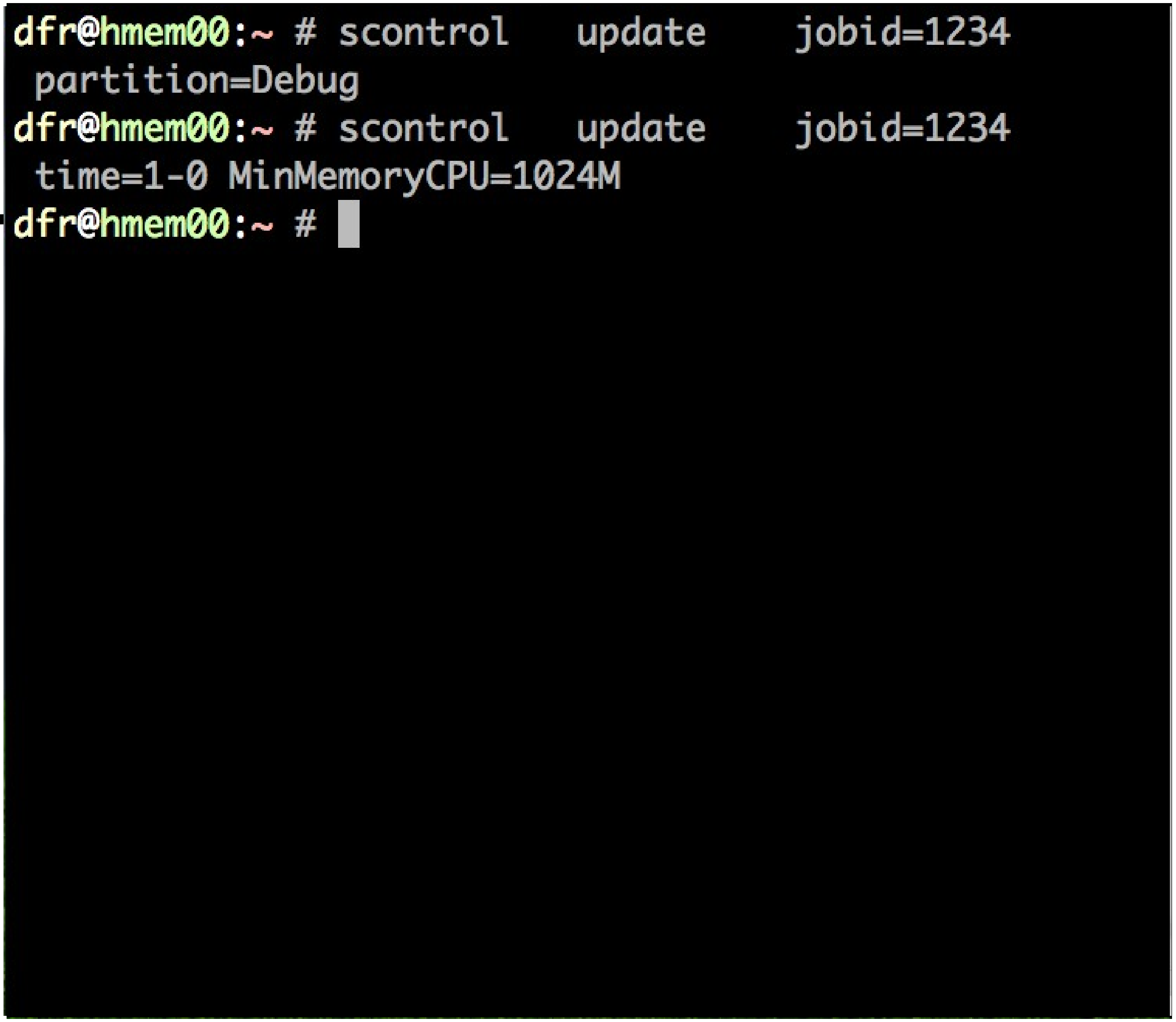
- scancel
- scontrol
- sview



```
SCONTROL(1)                    Slurm components
              SCONTROL(1)

NAME
       scontrol  -  Used  view and modify Slurm
       configuration and state.

SYNOPSIS
       scontrol [OPTIONS...] [COMMAND...]

DESCRIPTION
       scontrol is used to view or modify Slurm
       configuration  including: job, job step,
       node, partition, reservation, and  over-
       all  system  configuration.  Most of the
       commands can only be  executed  by  user
       root.  If  an  attempt to view or modify
       configuration information is made by  an
       unauthorized user, an error message will
       be printed and the requested action will
:
```

5. Control your job

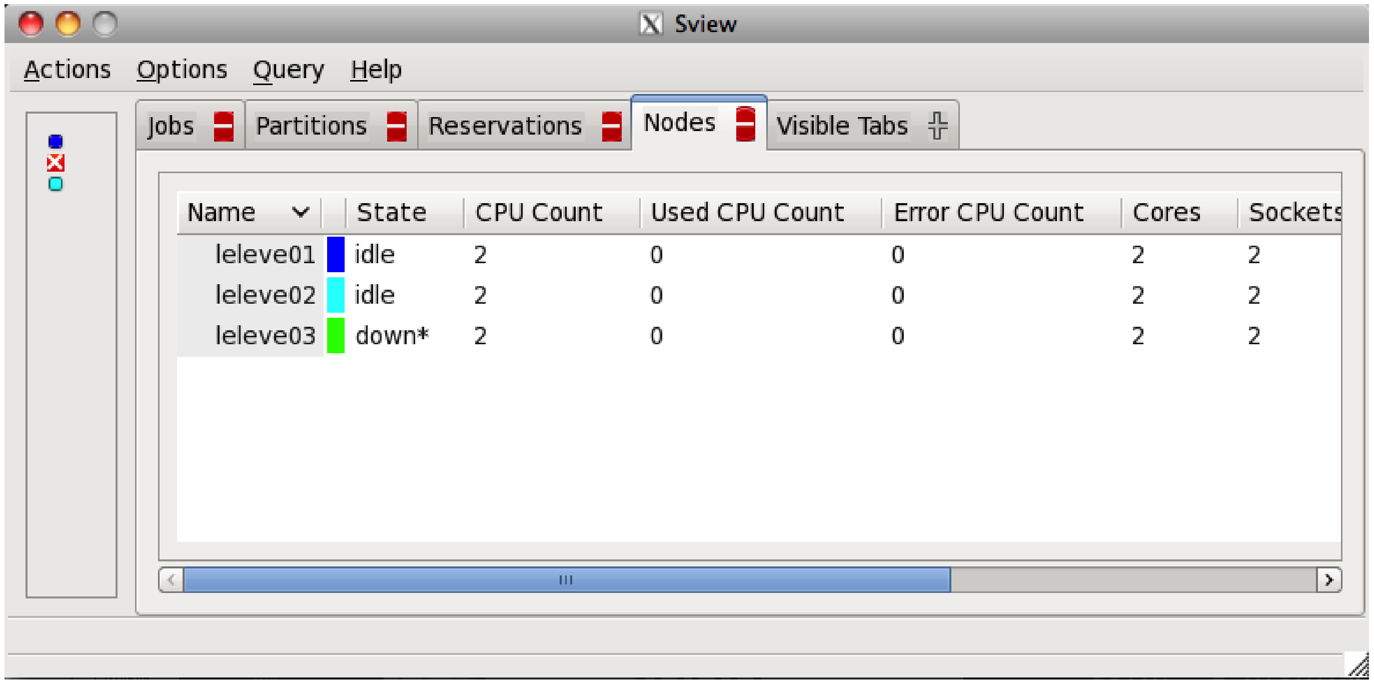
- scancel
- scontrol
- sview



```
dfr@hmem00:~ # scontrol   update    jobid=1234
 partition=Debug
dfr@hmem00:~ # scontrol   update    jobid=1234
 time=1-0 MinMemoryCPU=1024M
dfr@hmem00:~ #
```

5. Control your job

- scancel
- scontrol
- sview →

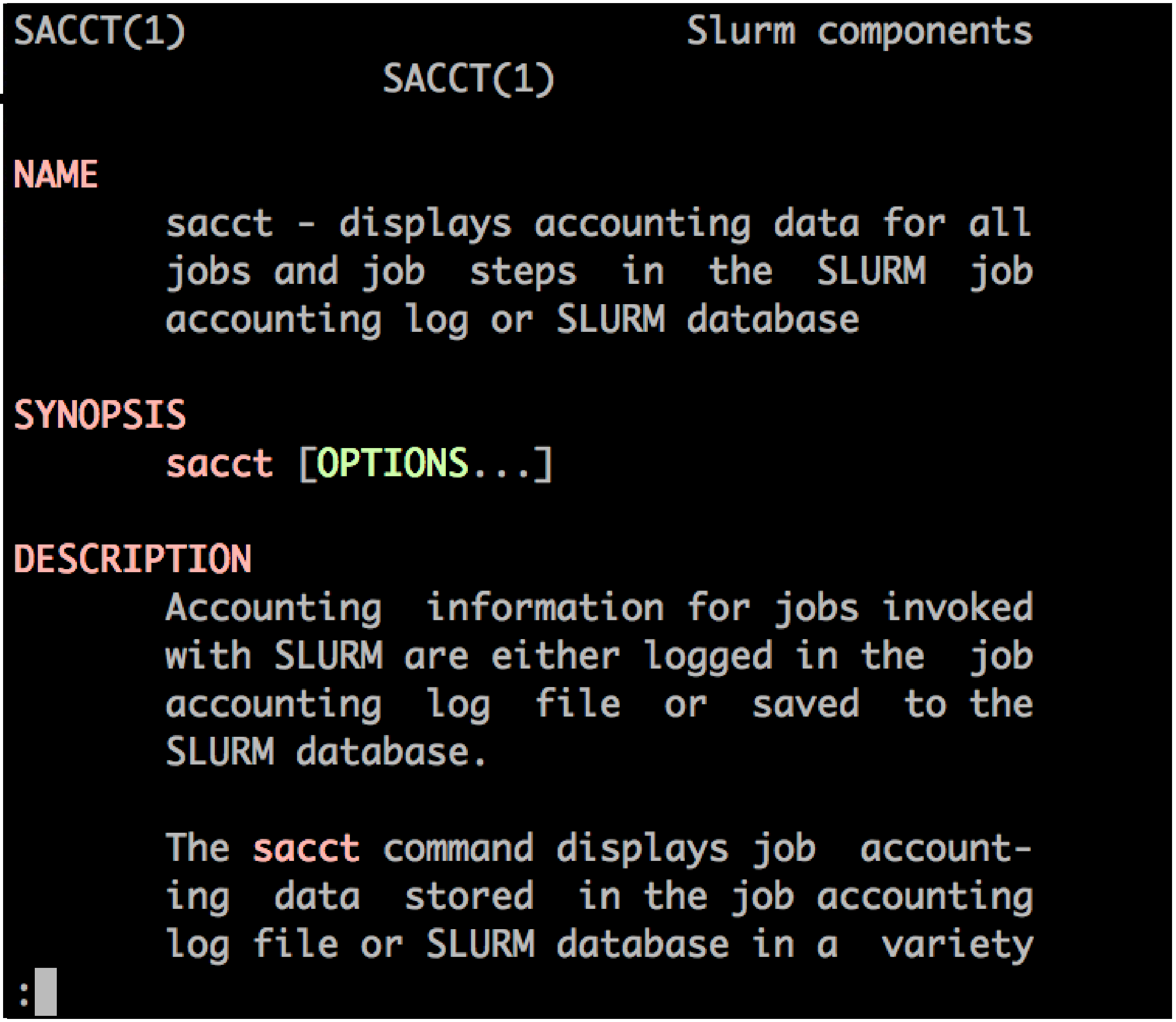


Name		State	CPU Count	Used CPU Count	Error CPU Count	Cores	Sockets
leleve01	🟦	idle	2	0	0	2	2
leleve02	🟦	idle	2	0	0	2	2
leleve03	🟩	down*	2	0	0	2	2

http://www.schedmd.com/slurmdocs/slurm_ug_2011/sview-users-guide.pdf

6. Job accounting

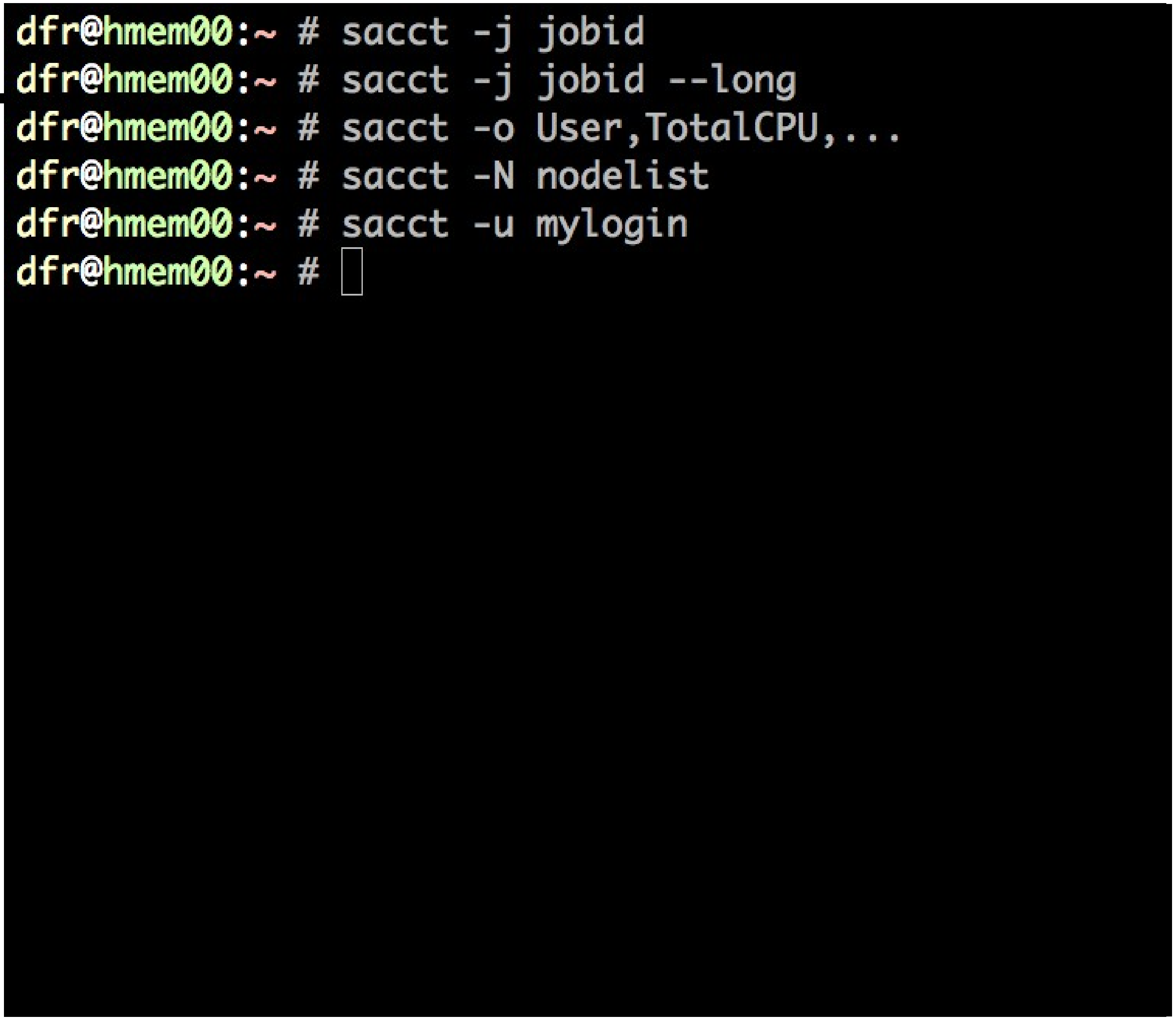
- sacct
- sreport
- sshare



```
SACCT(1)                     Slurm components
                   SACCT(1)

NAME
       sacct - displays accounting data for all
       jobs and job  steps  in  the  SLURM  job
       accounting log or SLURM database

SYNOPSIS
       sacct [OPTIONS...]

DESCRIPTION
       Accounting  information for jobs invoked
       with SLURM are either logged in the  job
       accounting  log  file  or  saved  to the
       SLURM database.

       The sacct command displays job  account-
       ing  data  stored  in the job accounting
       log file or SLURM database in a  variety
:
```

6. Job accounting

- sacct
- sreport
- sshare



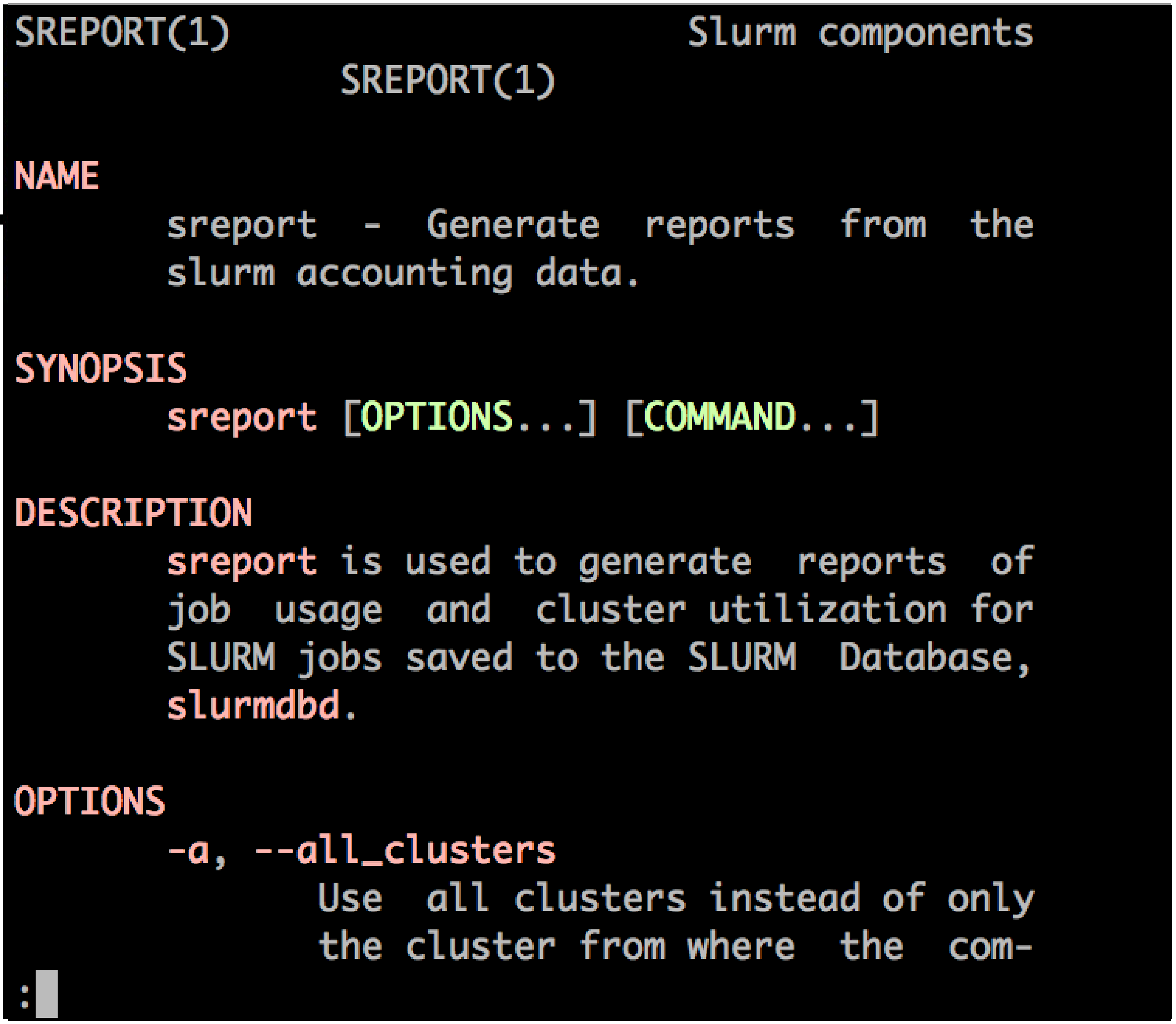
```
dfr@hmem00:~ # sacct -j jobid
dfr@hmem00:~ # sacct -j jobid --long
dfr@hmem00:~ # sacct -o User,TotalCPU,...
dfr@hmem00:~ # sacct -N nodelist
dfr@hmem00:~ # sacct -u mylogin
dfr@hmem00:~ # 
```

Look at your jobs

1. Connect to a cluster
2. run the “sacct” command to see your job history

6. Job accounting

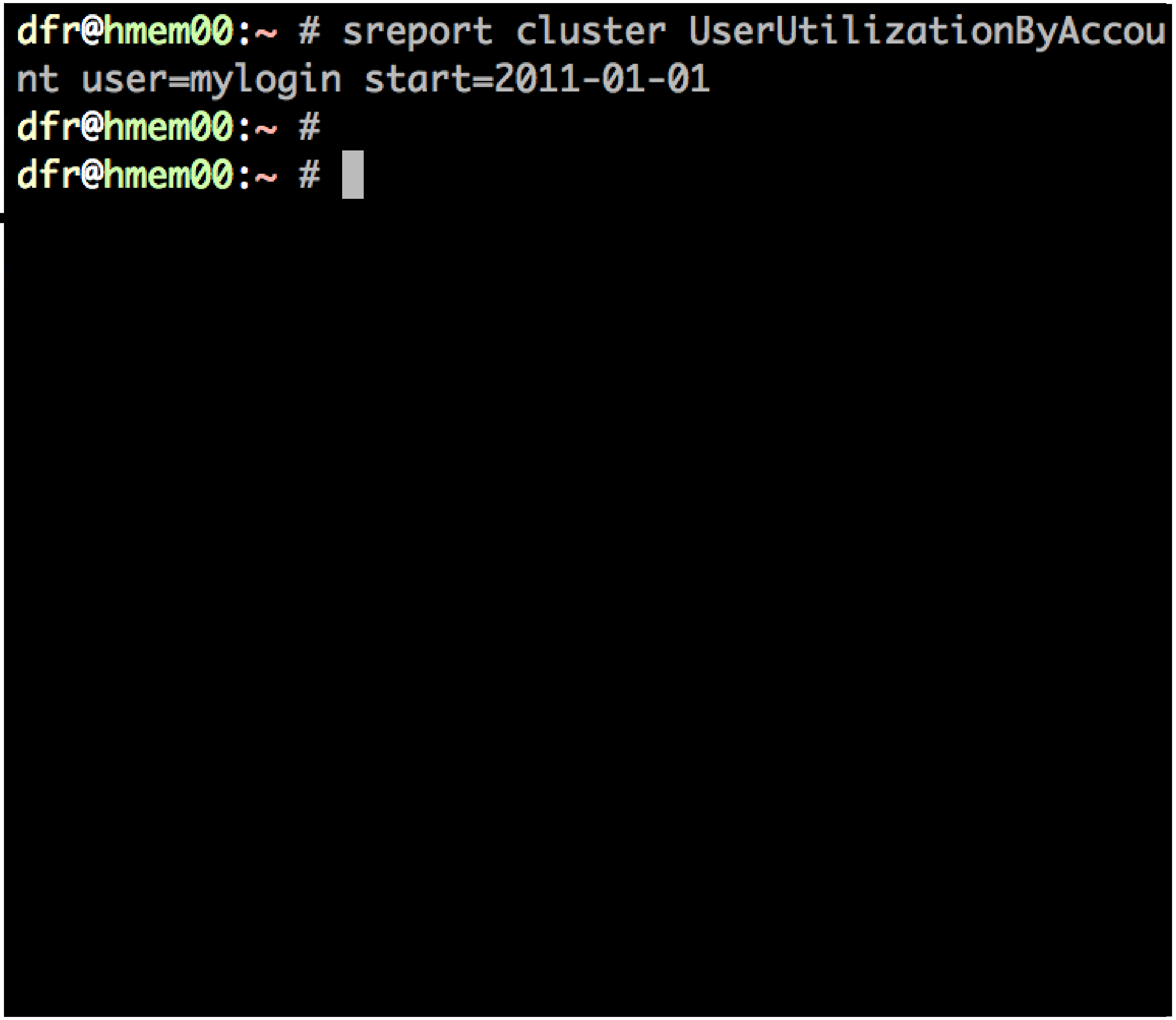
- sacct
- sreport
- sshare



```
SREPORT(1)                     Slurm components
                SREPORT(1)

NAME
       sreport  -  Generate  reports  from  the
       slurm accounting data.

SYNOPSIS
       sreport [OPTIONS...] [COMMAND...]

DESCRIPTION
       sreport is used to generate  reports  of
       job  usage  and  cluster utilization for
       SLURM jobs saved to the SLURM  Database,
       slurmdbd.

OPTIONS
       -a, --all_clusters
              Use  all clusters instead of only
              the cluster from where  the  com-
:
```

6. Job accounting

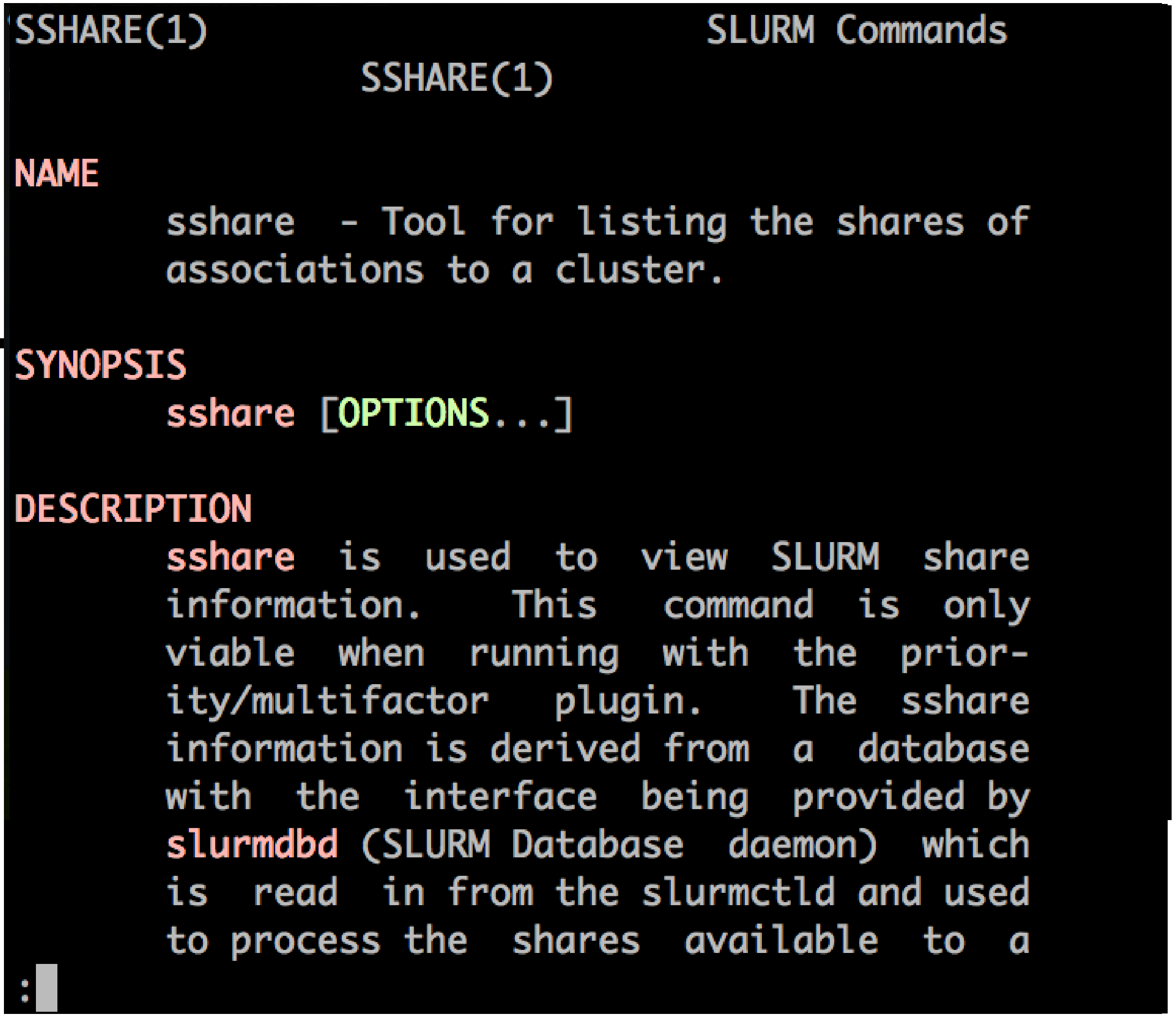
- sacct
- sreport
- sshare



```
dfr@hmem00:~ # sreport cluster UserUtilizationByAccou
nt user=mylogin start=2011-01-01
dfr@hmem00:~ #
dfr@hmem00:~ #
```

6. Job accounting

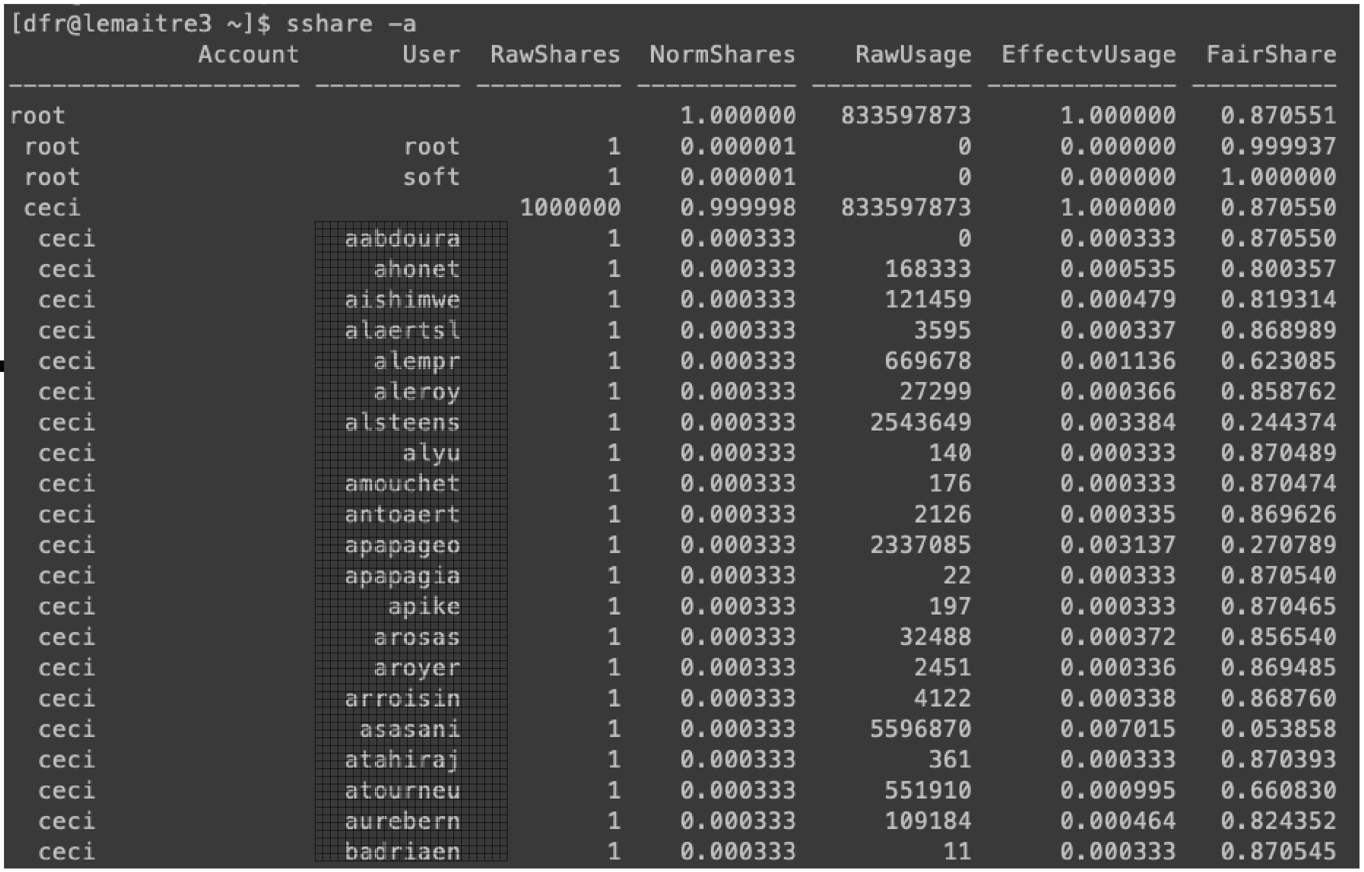
- sacct
- sreport
- sshare



```
SSHARE(1)                        SLURM Commands
              SSHARE(1)

NAME
       sshare  - Tool for listing the shares of
       associations to a cluster.

SYNOPSIS
       sshare [OPTIONS...]

DESCRIPTION
       sshare  is  used  to  view  SLURM  share
       information.   This   command  is  only
       viable  when  running  with  the  prior-
       ity/multifactor   plugin.    The  sshare
       information is derived from  a  database
       with  the  interface  being  provided by
       slurmdbd (SLURM Database  daemon)  which
       is  read  in from the slurmctld and used
       to process the  shares  available  to  a
:
```

6. Job accounting

- sacct
- sreport
- sshare

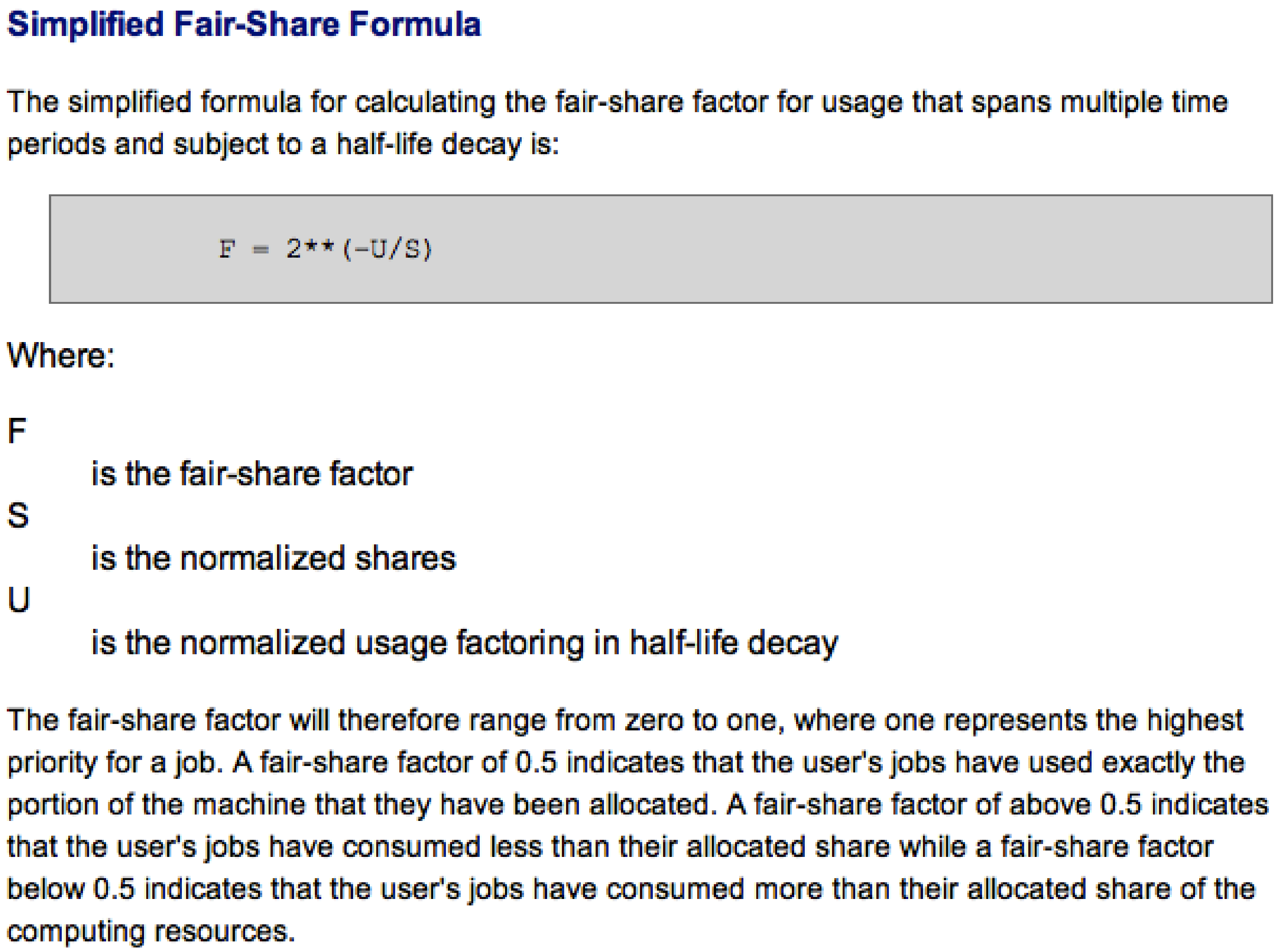


```
[dfr@lemaitre3 ~]$ sshare -a
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
root                                          1.000000   833597873      1.000000   0.870551
 root                      root          1    0.000001           0      0.000000   0.999937
 root                      soft          1    0.000001           0      0.000000   1.000000
 ceci                               1000000   0.999998   833597873      1.000000   0.870550
  ceci                 aabdoura          1    0.000333           0      0.000333   0.870550
  ceci                  ahonet           1    0.000333      168333      0.000535   0.800357
  ceci                 aishimwe          1    0.000333      121459      0.000479   0.819314
  ceci                 alaertsl          1    0.000333        3595      0.000337   0.868989
  ceci                   alempr          1    0.000333      669678      0.001136   0.623085
  ceci                   aleroy          1    0.000333       27299      0.000366   0.858762
  ceci                 alsteens          1    0.000333     2543649      0.003384   0.244374
  ceci                     alyu          1    0.000333         140      0.000333   0.870489
  ceci                 amouchet          1    0.000333         176      0.000333   0.870474
  ceci                 antoaert          1    0.000333        2126      0.000335   0.869626
  ceci                 apapageo          1    0.000333     2337085      0.003137   0.270789
  ceci                 apapagia          1    0.000333          22      0.000333   0.870540
  ceci                    apike          1    0.000333         197      0.000333   0.870465
  ceci                   arosas          1    0.000333       32488      0.000372   0.856540
  ceci                   aroyer          1    0.000333        2451      0.000336   0.869485
  ceci                 arroisin          1    0.000333        4122      0.000338   0.868760
  ceci                  asasani          1    0.000333     5596870      0.007015   0.053858
  ceci                 atahiraj          1    0.000333         361      0.000333   0.870393
  ceci                 atourneu          1    0.000333      551910      0.000995   0.660830
  ceci                 aurebern          1    0.000333      109184      0.000464   0.824352
  ceci                 badriaen          1    0.000333          11      0.000333   0.870545
```

The rules of fairshare

- Fairshare directly influences job priority
- A share is allocated to you: 1/#users
- If your actual usage is above that share, your fairshare value is decreased towards 0.
- If your actual usage is below that share, your fairshare value is increased towards 1.
- The actual usage taken into account decreases over time; usage two months ago has less impact on the fairshare than usage two days ago.

A word about fairshare



Simplified Fair-Share Formula

The simplified formula for calculating the fair-share factor for usage that spans multiple time periods and subject to a half-life decay is:

```
F = 2**(-U/S)
```

Where:

F
: is the fair-share factor

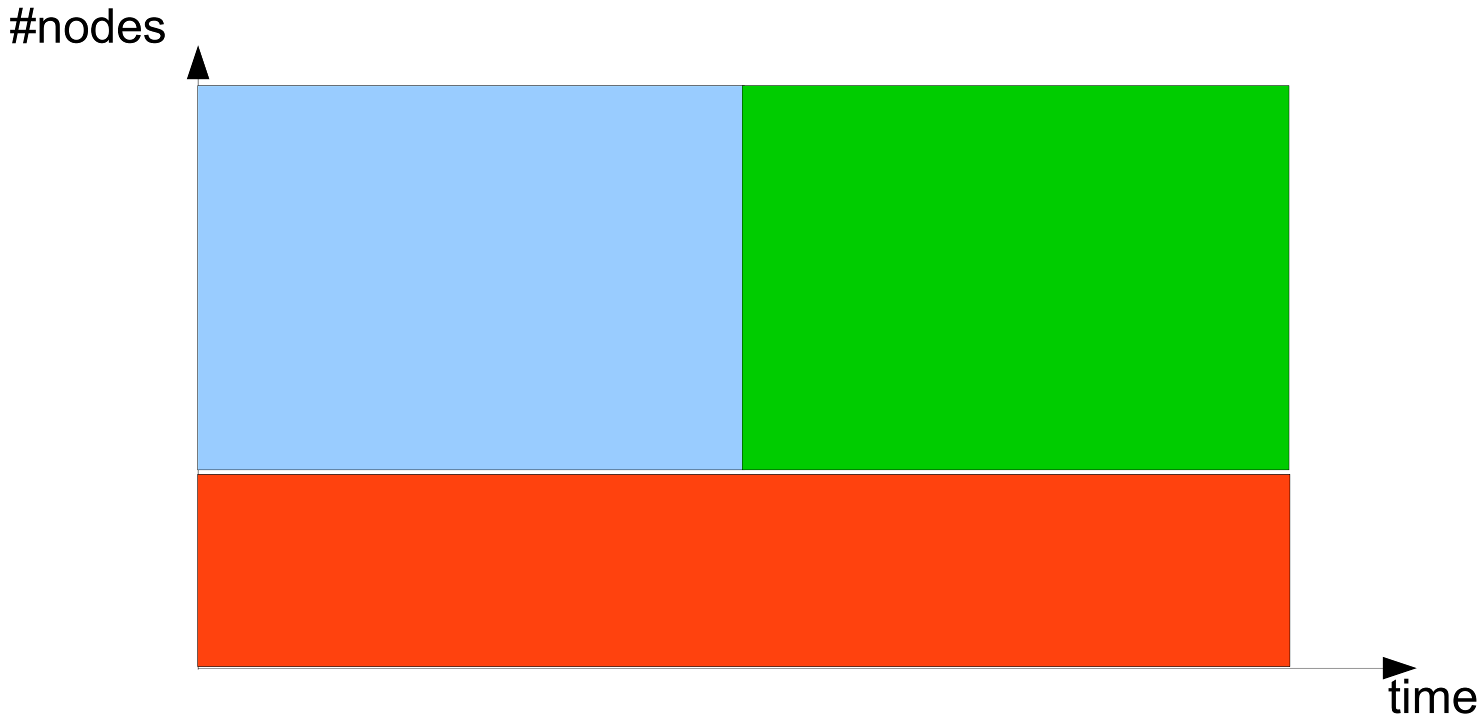
S
: is the normalized shares

U
: is the normalized usage factoring in half-life decay

The fair-share factor will therefore range from zero to one, where one represents the highest priority for a job. A fair-share factor of 0.5 indicates that the user's jobs have used exactly the portion of the machine that they have been allocated. A fair-share factor of above 0.5 indicates that the user's jobs have consumed less than their allocated share while a fair-share factor below 0.5 indicates that the user's jobs have consumed more than their allocated share of the computing resources.

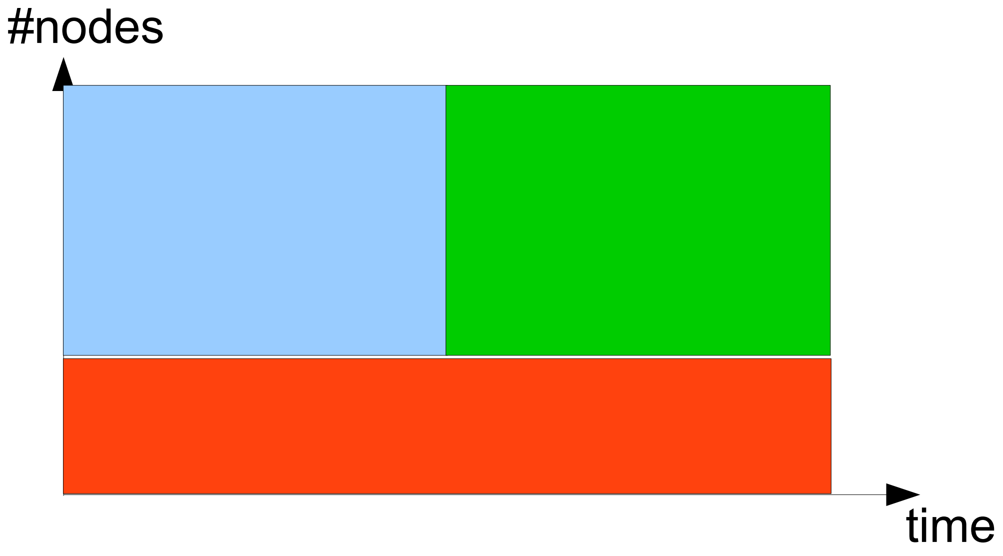
A word about fairshare

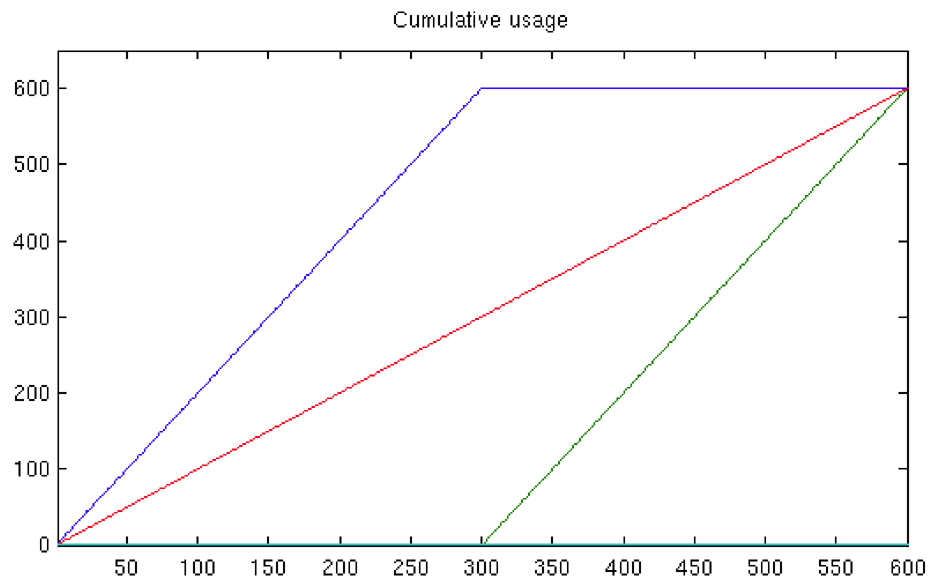
- Assume 3 users, 3-cores cluster
  - Red uses 1 core for a certain period of time
  - Blue uses 2 cores for half that period
  - Red uses 2 cores afterwards



A word about fairshare

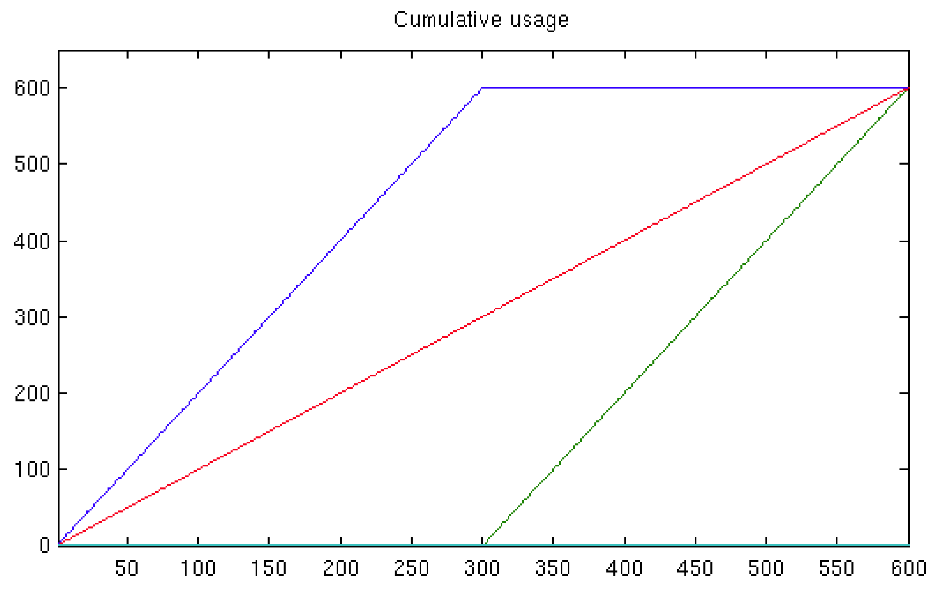
- Assume 3 users, 3-cores cluster
  - Red uses 1 core for a certain period of time
  - Blue uses 2 cores for half that period
  - Red uses 2 cores afterwards

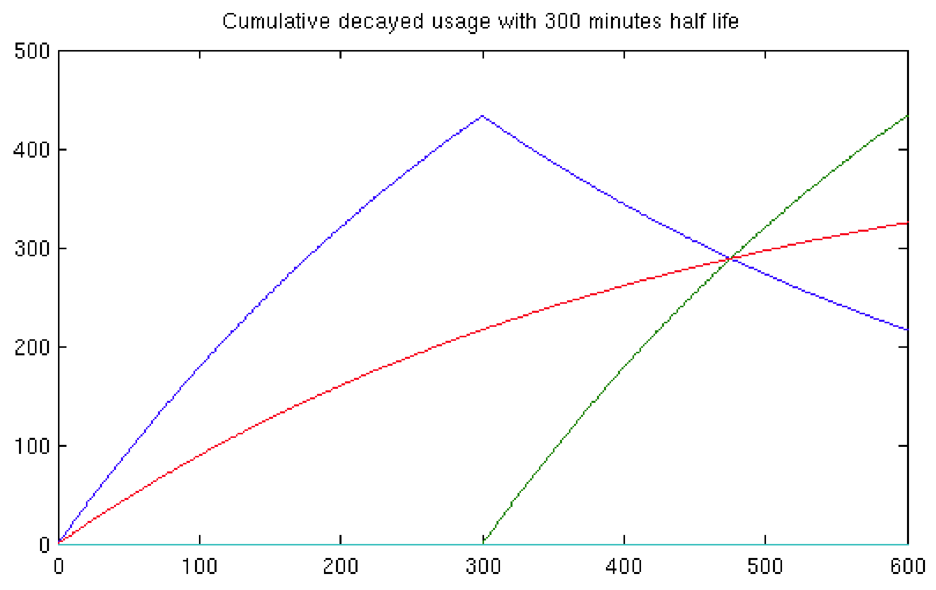




A word about fairshare

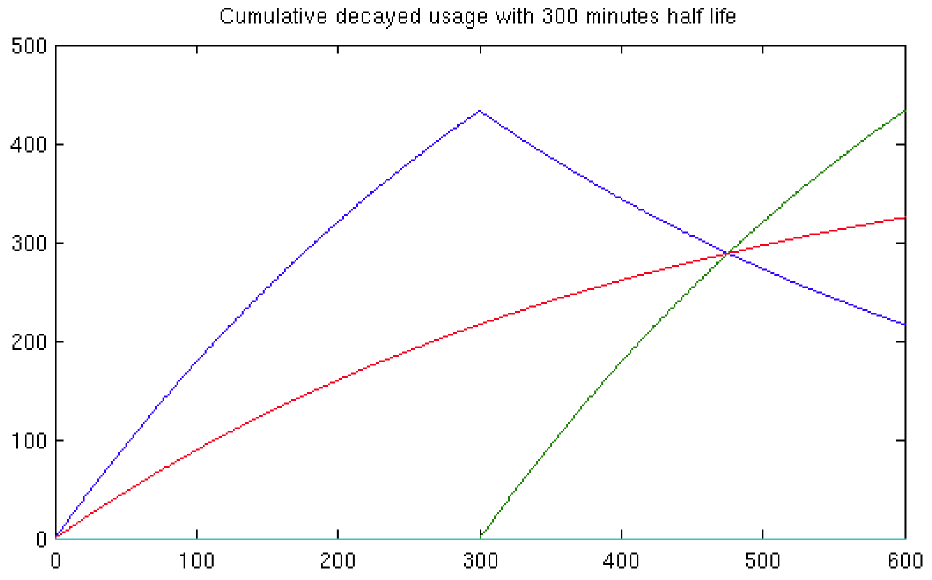
- Assume 3 users, 3-cores cluster
  - Red uses 1 core for a certain period of time
  - Blue uses 2 cores for half that period
  - Red uses 2 cores afterwards

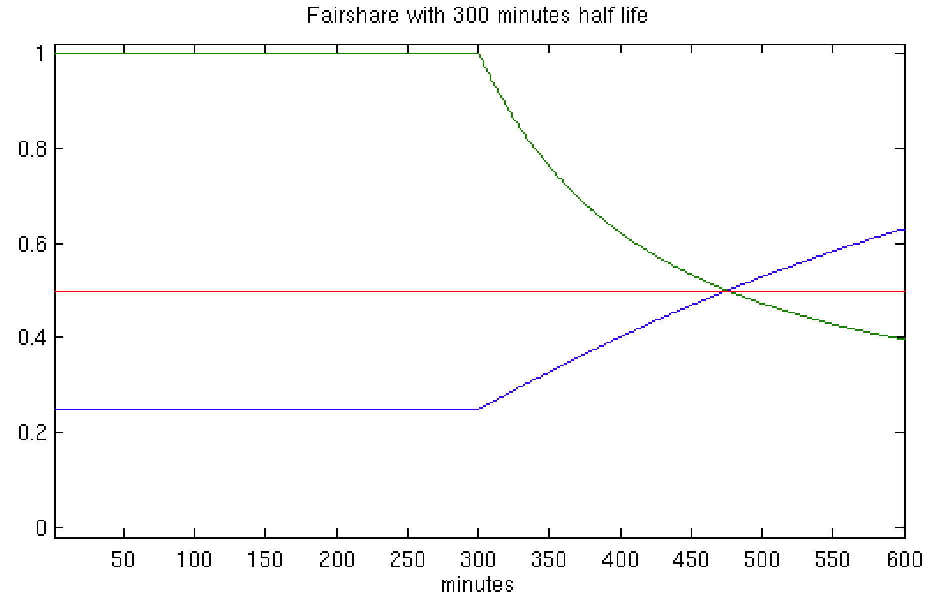


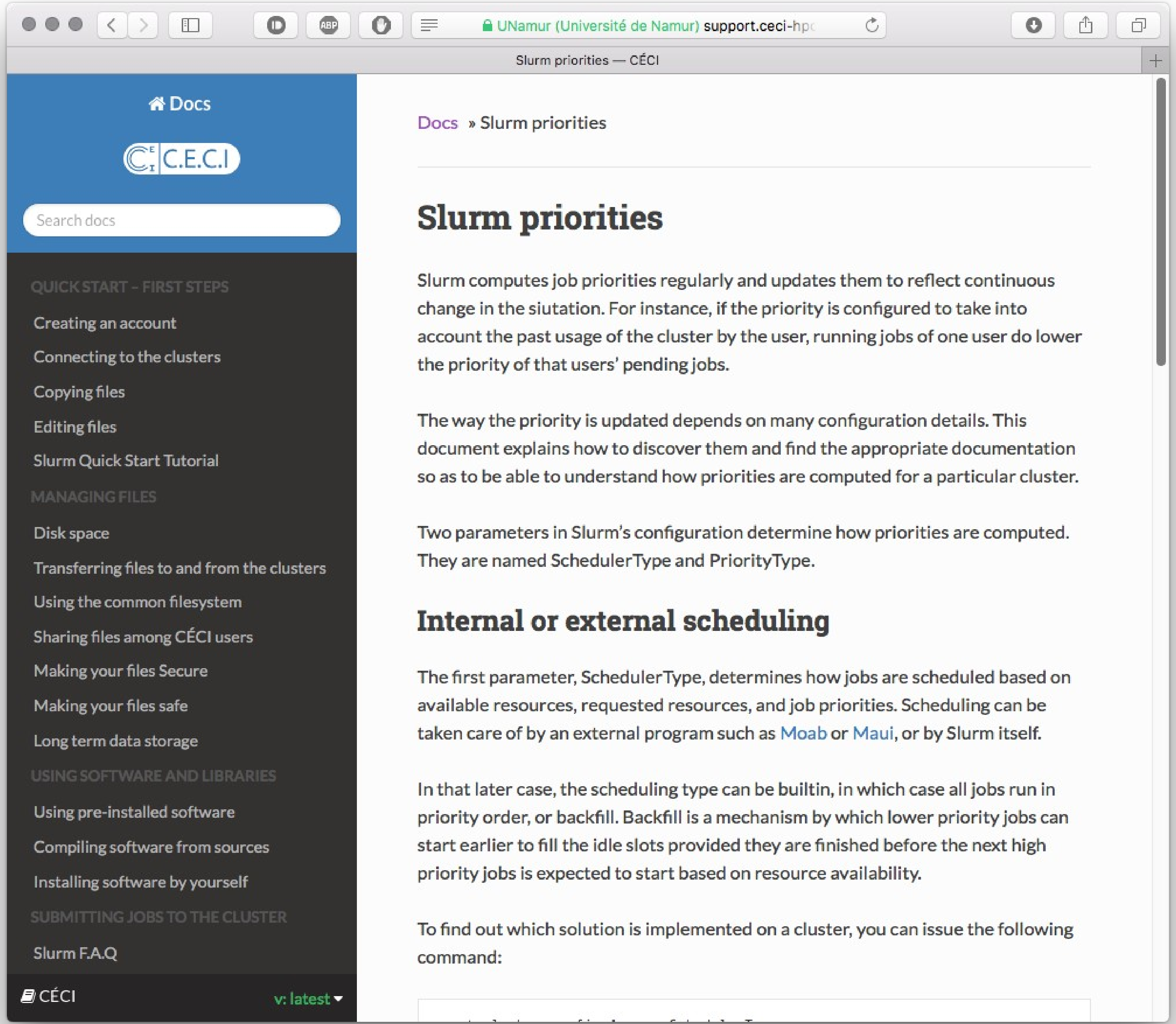


A word about fairshare

- Assume 3 users, 3-cores cluster
  - Red uses 1 core for a certain period of time
  - Blue uses 2 cores for half that period
  - Red uses 2 cores afterwards







UNamur (Université de Namur) support.ceci-hpc

Slurm priorities — CÉCI

Docs

C.E.C.I

Search docs

QUICK START – FIRST STEPS
- Creating an account
- Connecting to the clusters
- Copying files
- Editing files
- Slurm Quick Start Tutorial

MANAGING FILES
- Disk space
- Transferring files to and from the clusters
- Using the common filesystem
- Sharing files among CÉCI users
- Making your files Secure
- Making your files safe
- Long term data storage

USING SOFTWARE AND LIBRARIES
- Using pre-installed software
- Compiling software from sources
- Installing software by yourself

SUBMITTING JOBS TO THE CLUSTER
- Slurm F.A.Q

CÉCI v: latest

Docs » Slurm priorities

Slurm priorities

Slurm computes job priorities regularly and updates them to reflect continuous change in the siutation. For instance, if the priority is configured to take into account the past usage of the cluster by the user, running jobs of one user do lower the priority of that users' pending jobs.

The way the priority is updated depends on many configuration details. This document explains how to discover them and find the appropriate documentation so as to be able to understand how priorities are computed for a particular cluster.

Two parameters in Slurm's configuration determine how priorities are computed. They are named SchedulerType and PriorityType.

Internal or external scheduling

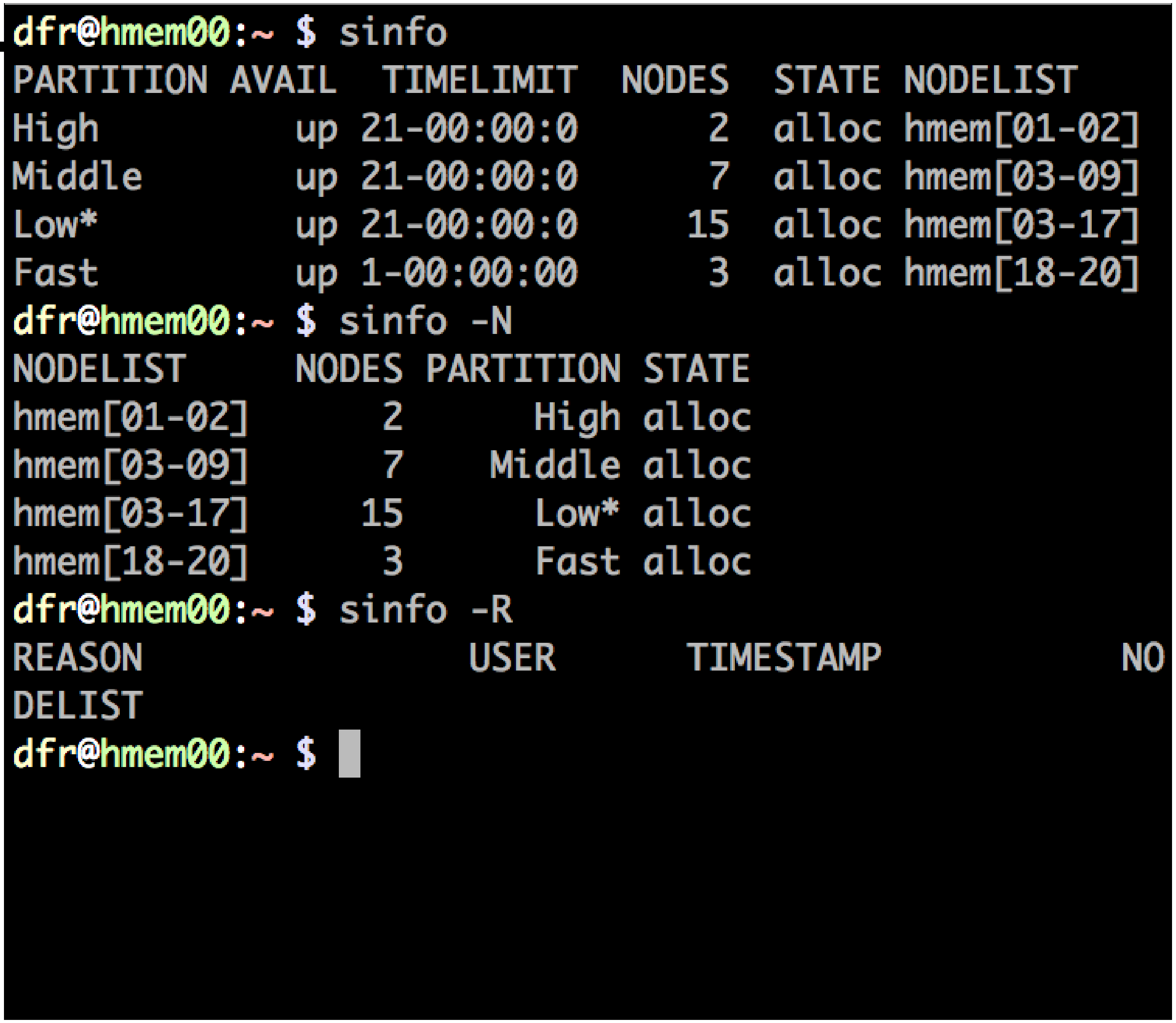
The first parameter, SchedulerType, determines how jobs are scheduled based on available resources, requested resources, and job priorities. Scheduling can be taken care of by an external program such as Moab or Maui, or by Slurm itself.

In that later case, the scheduling type can be builtin, in which case all jobs run in priority order, or backfill. Backfill is a mechanism by which lower priority jobs can start earlier to fill the idle slots provided they are finished before the next high priority jobs is expected to start based on resource availability.

To find out which solution is implemented on a cluster, you can issue the following command:

Getting cluster info

- sinfo



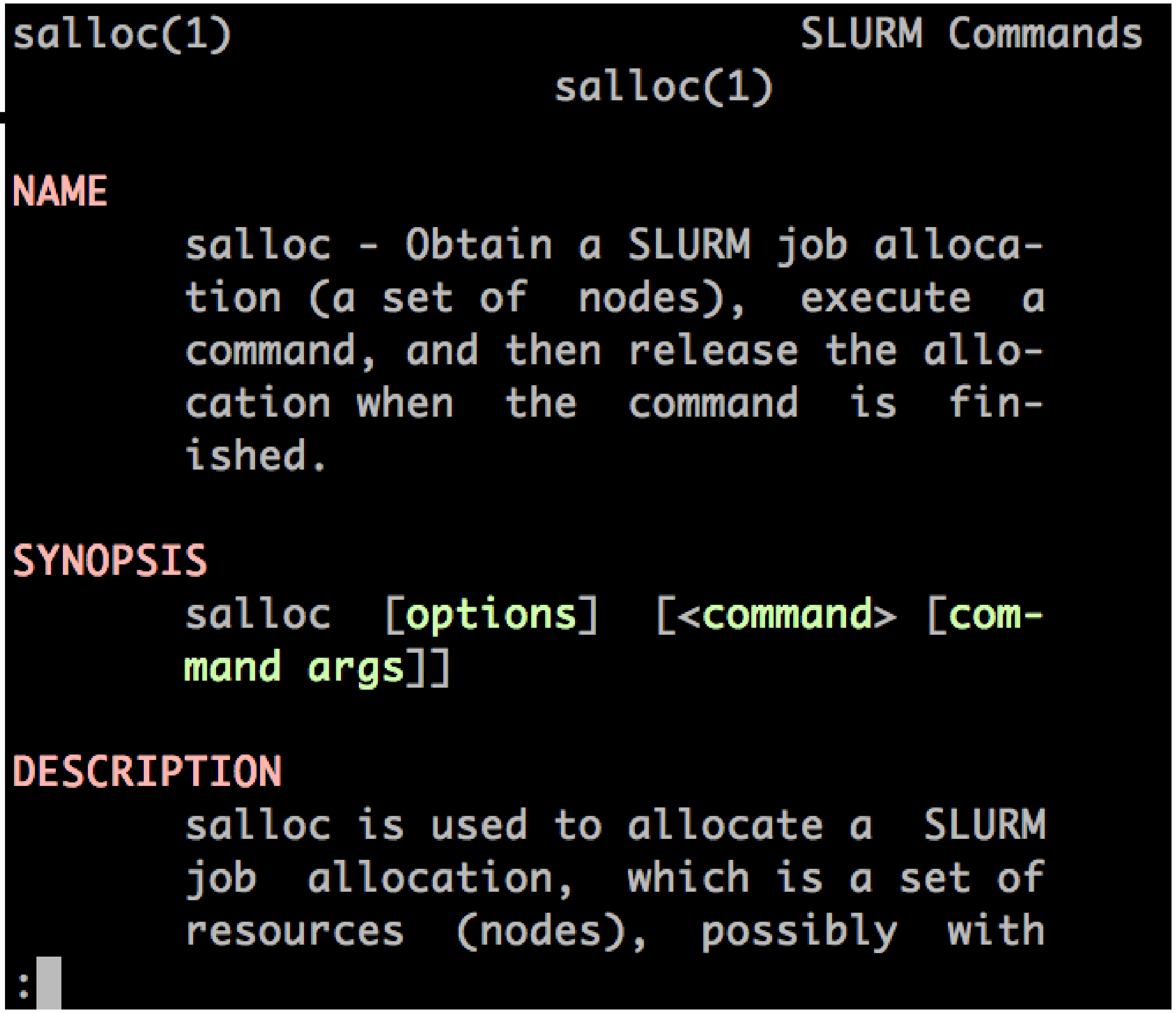
```
dfr@hmem00:~ $ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
High         up 21-00:00:0      2  alloc hmem[01-02]
Middle       up 21-00:00:0      7  alloc hmem[03-09]
Low*         up 21-00:00:0     15  alloc hmem[03-17]
Fast         up 1-00:00:00      3  alloc hmem[18-20]
dfr@hmem00:~ $ sinfo -N
NODELIST     NODES PARTITION STATE
hmem[01-02]      2      High alloc
hmem[03-09]      7    Middle alloc
hmem[03-17]     15      Low* alloc
hmem[18-20]      3      Fast alloc
dfr@hmem00:~ $ sinfo -R
REASON               USER      TIMESTAMP           NO
DELIST
dfr@hmem00:~ $ 
```

Get the cluster info

1. Connect to a cluster
2. run the “sinfo” command
3. run the “sinfo –Nl” command
4. run the “sinfo ––clusters all” command
5. run the “sacct ––federation” command

Interactive work

- salloc

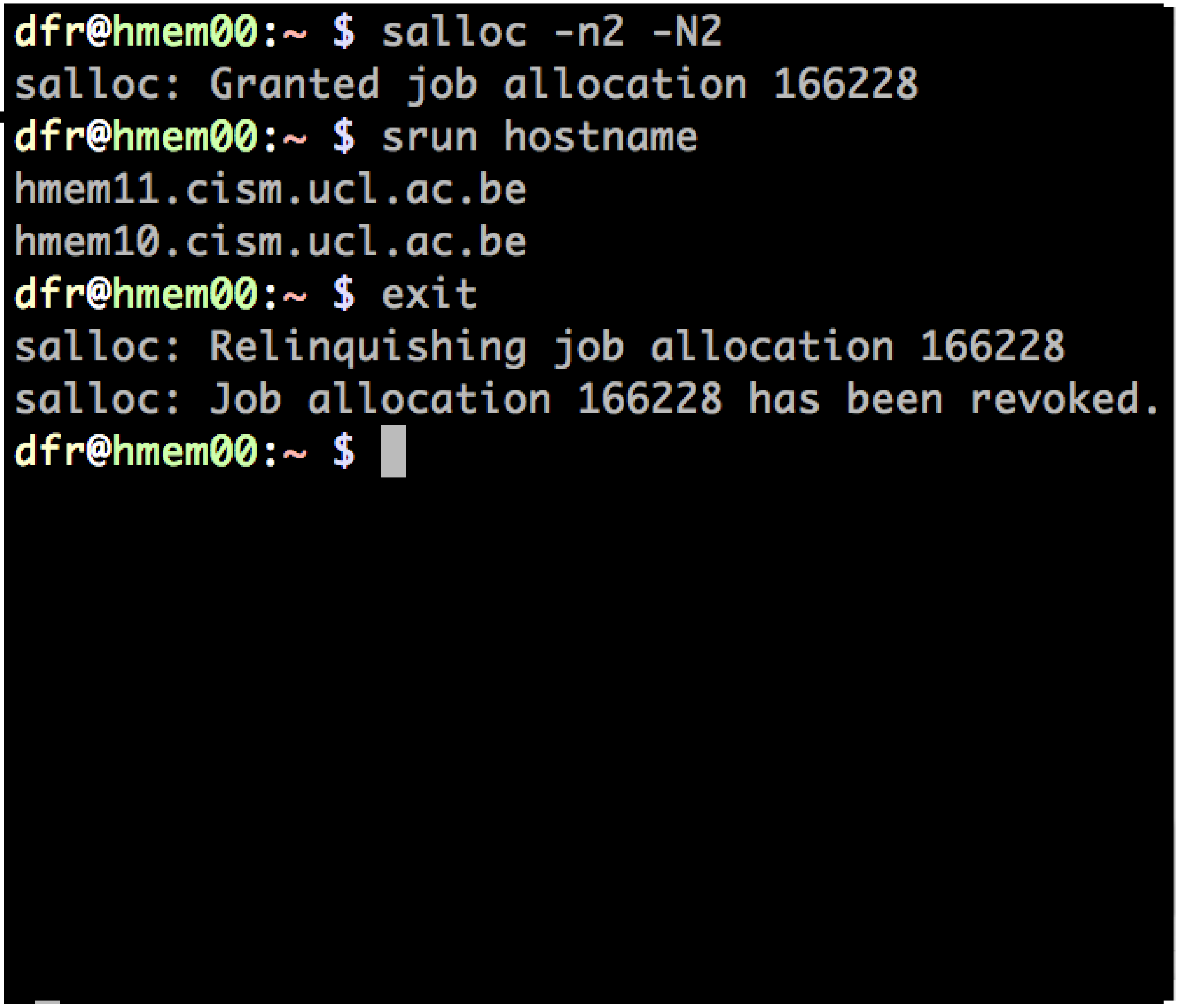


```
salloc(1)                              SLURM Commands
                     salloc(1)

NAME
       salloc - Obtain a SLURM job alloca-
       tion (a set of  nodes),  execute  a
       command, and then release the allo-
       cation when  the  command  is  fin-
       ished.

SYNOPSIS
       salloc  [options]  [<command> [com-
       mand args]]

DESCRIPTION
       salloc is used to allocate a  SLURM
       job  allocation,  which is a set of
       resources  (nodes),  possibly  with
:
```

salloc --ntasks=4 --nodes=2

Interactive work

- salloc

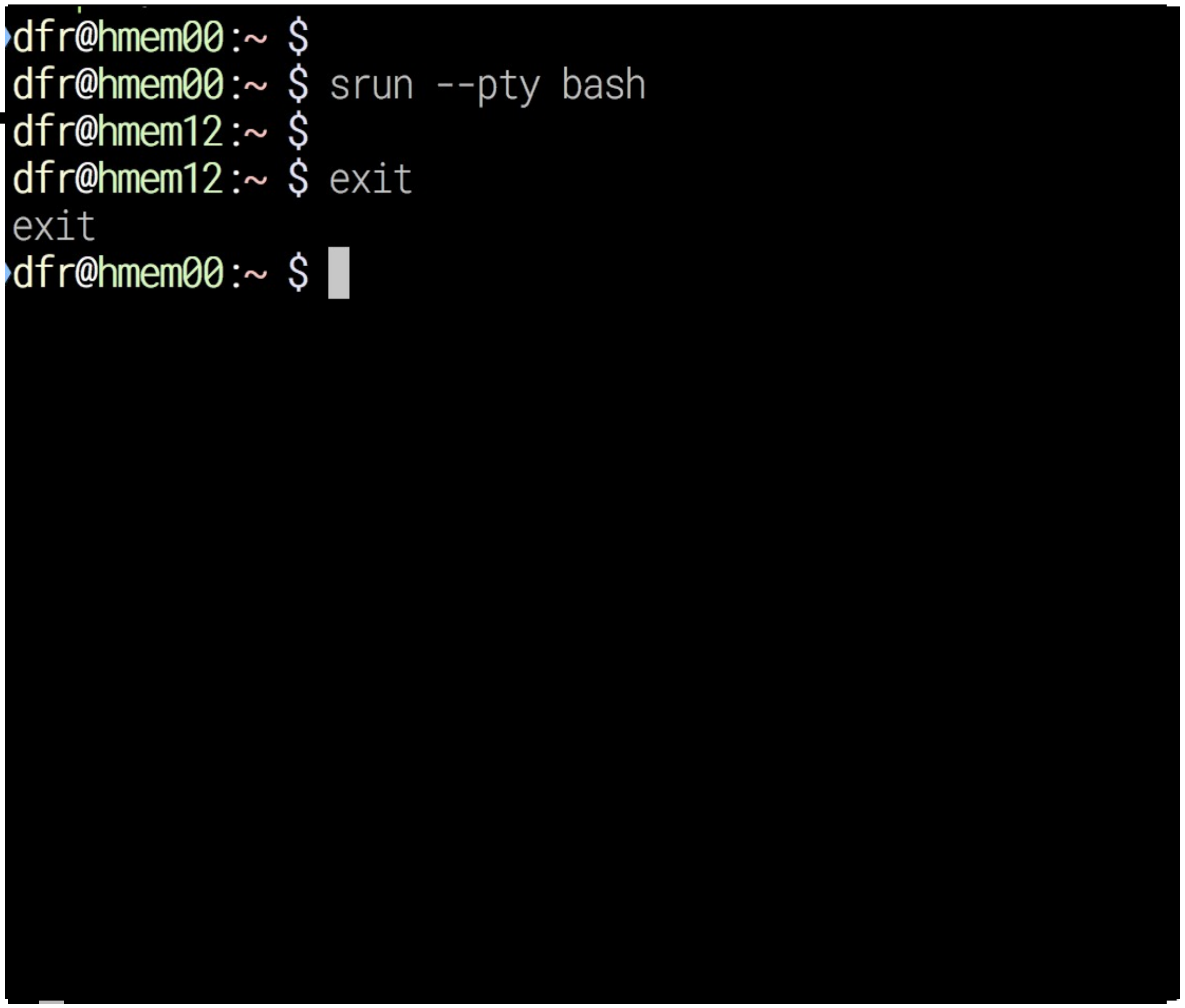


```
dfr@hmem00:~ $ salloc -n2 -N2
salloc: Granted job allocation 166228
dfr@hmem00:~ $ srun hostname
hmem11.cism.ucl.ac.be
hmem10.cism.ucl.ac.be
dfr@hmem00:~ $ exit
salloc: Relinquishing job allocation 166228
salloc: Job allocation 166228 has been revoked.
dfr@hmem00:~ $
```

salloc --ntasks=4 --nodes=2

Interactive work

- srun



```
dfr@hmem00:~ $
dfr@hmem00:~ $ srun --pty bash
dfr@hmem12:~ $
dfr@hmem12:~ $ exit
exit
dfr@hmem00:~ $
```

srun --pty bash

Summary

- Explore the environment
  - Get node features (sinfo --node --long)
  - Get node usage (sinfo --summarize)
- Submit a job:
  - Define the resource you need
  - Determine what the job should do

    } job script
  - Submit the job script (sbatch)
  - View the job status (squeue)
  - Get accounting information (sacct)

How to choose the number of CPUs, memory, and time?

CPU cores

Memory

Disk space

Network

Accelerators

Software

Licenses

Let

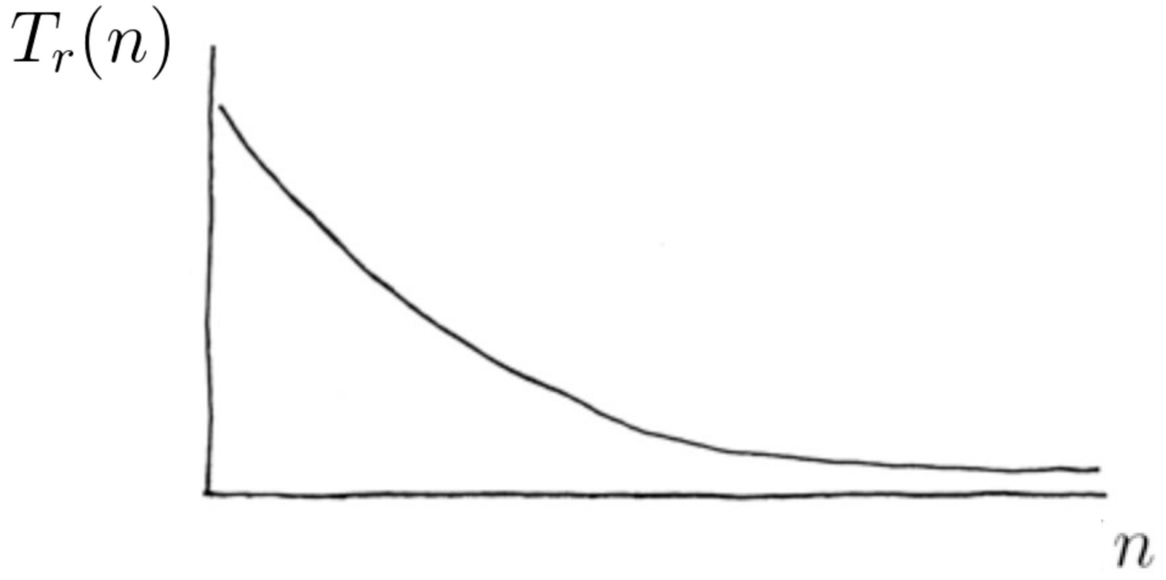
- *t* be the requested time,
- *m* the requested memory,
- *n* the requested number of CPUs, and
- ε the risk for your job to be killed due to limit trespassing

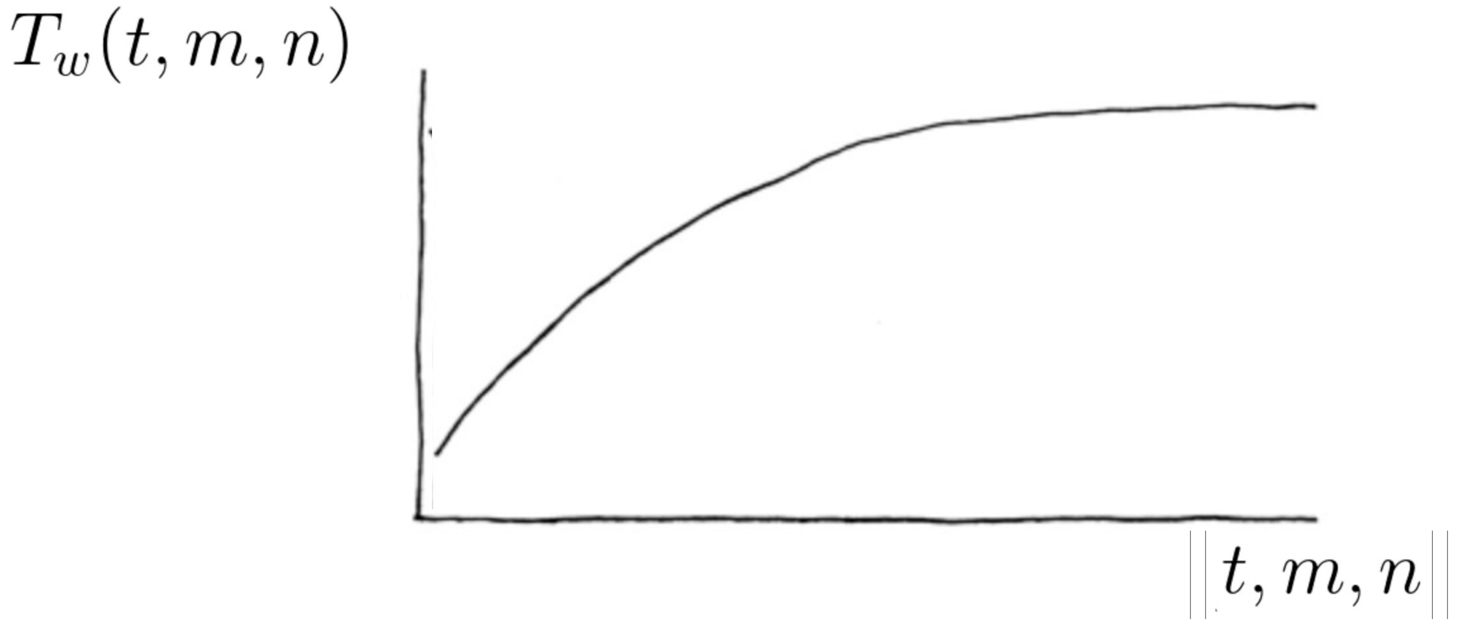
The problem is: $$\min_{t,m,n} T_w(t,m,n) + T_r(n)$$

subject to:

$$\mathsf{P}(T_r(n) > t) < \epsilon$$
$$\mathsf{P}(M_r(n) > m) < \epsilon$$

with	$T_w(t,m,n)$	the job waiting time in the queue
	$T_r(n)$	the job running time
	$M_r(n)$	the job memory usage


$T_r(n)$
n


$T_w(t, m, n)$
$\|t, m, n\|$

-N, **--nodes**=<*minnodes*[-*maxnodes*]>

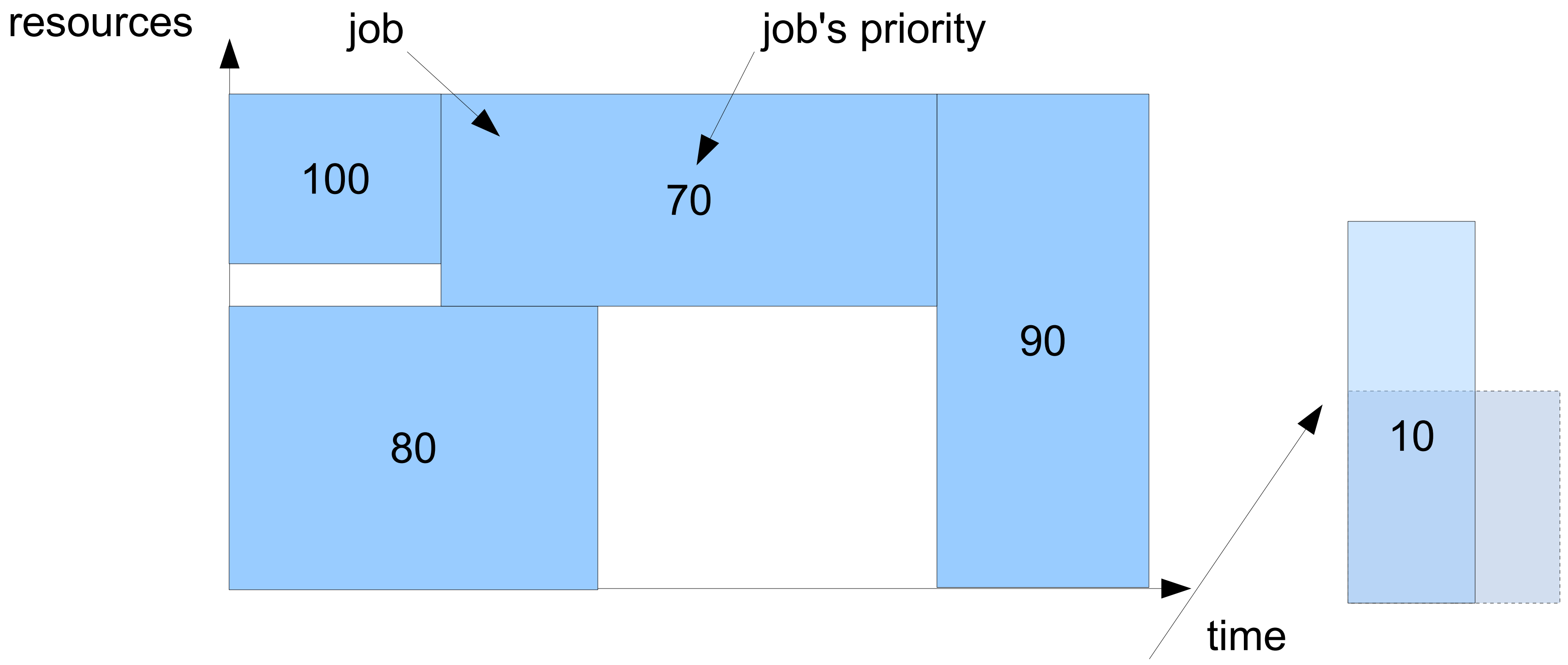
Request that a minimum of *minnodes* nodes be allocated to this job. A maximum node count may also be specified with *maxnodes*. If only one number is specified, this is used as both the minimum and maximum node count. The partition's node limits supersede those of the job. If a job's node limits are outside of the range permitted for its associated partition, the job will be left in a PENDING state. This permits possible execution at a later time, when the partition limit is changed. If a job node limit exceeds the number of nodes configured in the partition, the job will be rejected. Note that the environment variable **SLURM_NNODES** will be set to the count of nodes actually allocated to the job. See the **ENVIRONMENT VARIABLES** section for more information. If **-N** is not specified, the default behavior is to allocate enough nodes to satisfy the requirements of the **-n** and **-c** options. The job will be allocated as many nodes as possible within the range specified and without delaying the initiation of the job. The node count specification may include a numeric value followed by a suffix of "k" (multiplies numeric value by 1,024) or "m" (multiplies numeric value by 1,048,576).

--time-min=<*time*>

Set a minimum time limit on the job allocation. If specified, the job may have it's **--time** limit lowered to a value no lower than **--time-min** if doing so permits the job to begin execution earlier than otherwise possible. The job's time limit will not be changed after the job is allocated resources. This is performed by a backfill scheduling algorithm to allocate resources otherwise reserved for higher priority jobs. Acceptable time formats include "minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds".

A word about backfill

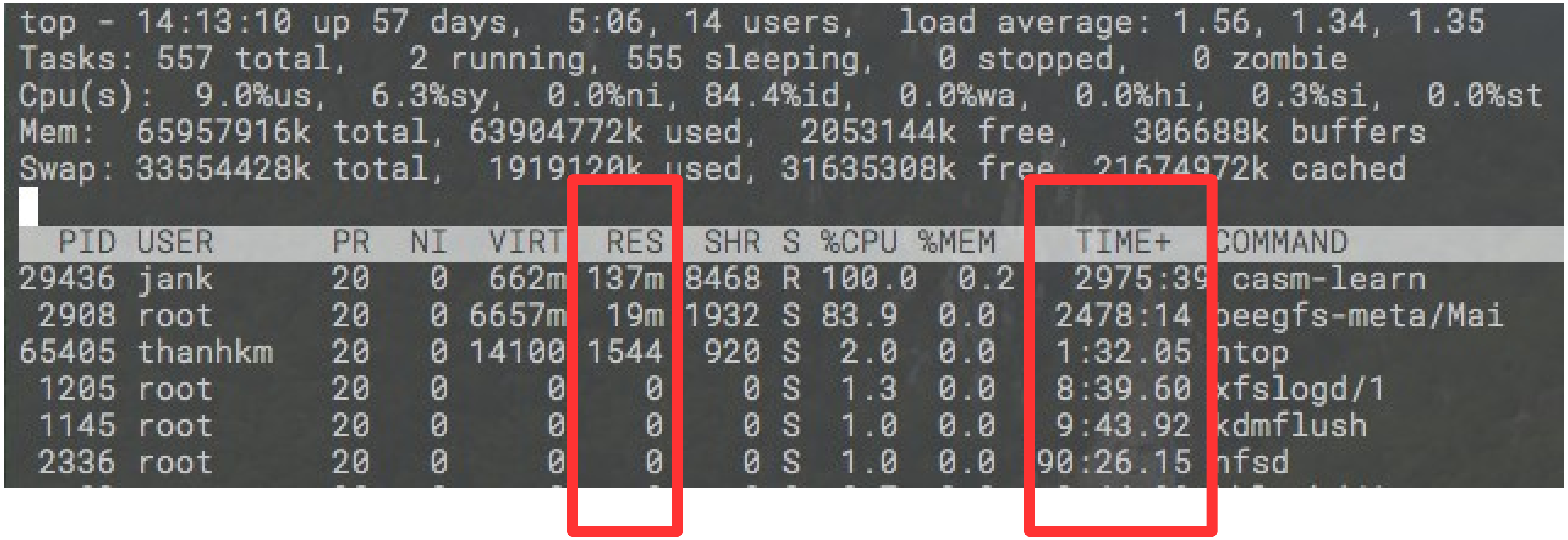
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



This job is flexible thanks to --nodes=16-24 and --time-min set to 60% of --time for instance

Practical approach

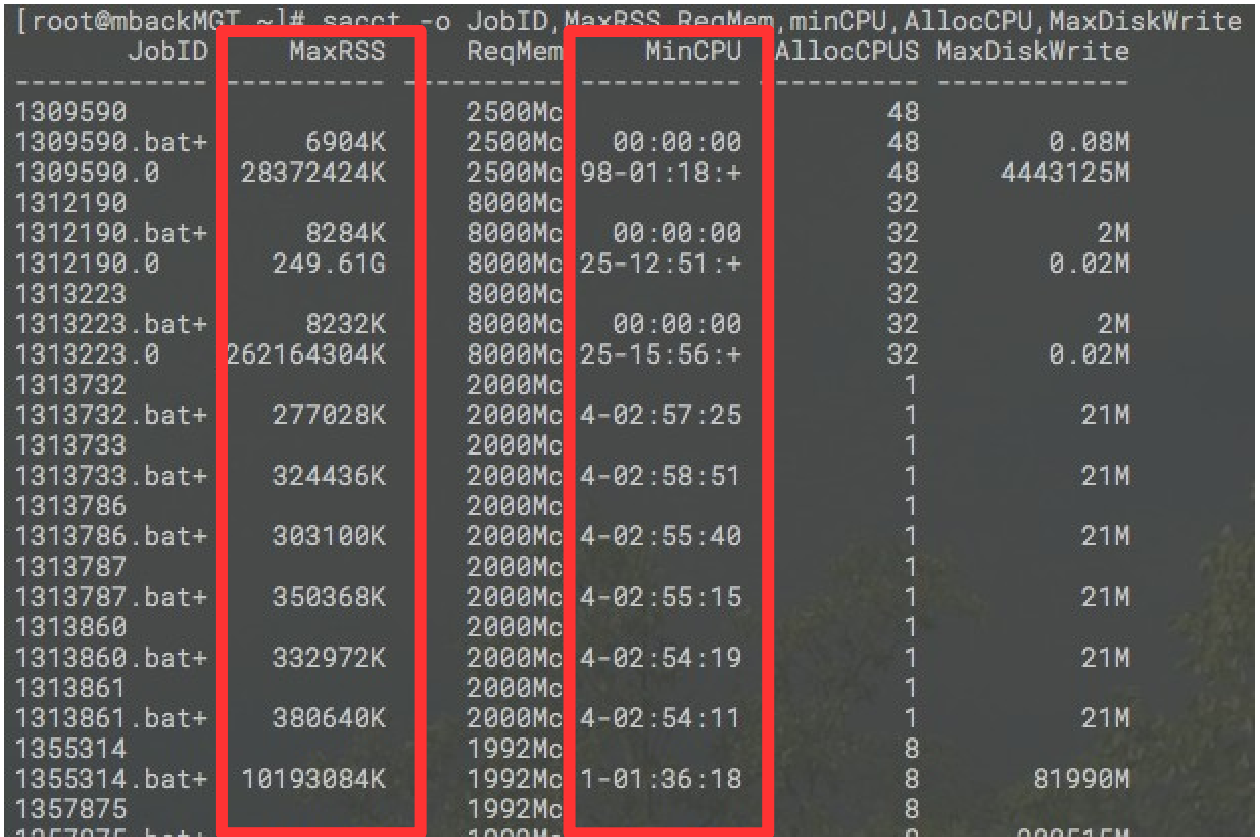
- Run a sized-down problem on your laptop and observe memory usage and time needed for several values of the number of CPUs for the first few iterations (`top`).
- Extrapolate for larger values of CPUs



```
top - 14:13:10 up 57 days,  5:06, 14 users,  load average: 1.56, 1.34, 1.35
Tasks: 557 total,   2 running, 555 sleeping,   0 stopped,   0 zombie
Cpu(s):  9.0%us,  6.3%sy,  0.0%ni, 84.4%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:  65957916k total, 63904772k used,  2053144k free,   306688k buffers
Swap: 33554428k total,  1919120k used, 31635308k free, 21674972k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29436 jank      20   0  662m 137m 8468 R 100.0  0.2  2975:39 casm-learn
 2908 root      20   0 6657m  19m 1932 S 83.9  0.0  2478:14 beegfs-meta/Mai
65405 thanhkm   20   0 14100 1544  920 S  2.0  0.0   1:32.05 htop
 1205 root      20   0     0    0    0 S  1.3  0.0   8:39.60 xfslogd/1
 1145 root      20   0     0    0    0 S  1.0  0.0   9:43.92 kdmflush
 2336 root      20   0     0    0    0 S  1.0  0.0  90:26.15 nfsd
```

Pragmatic approach

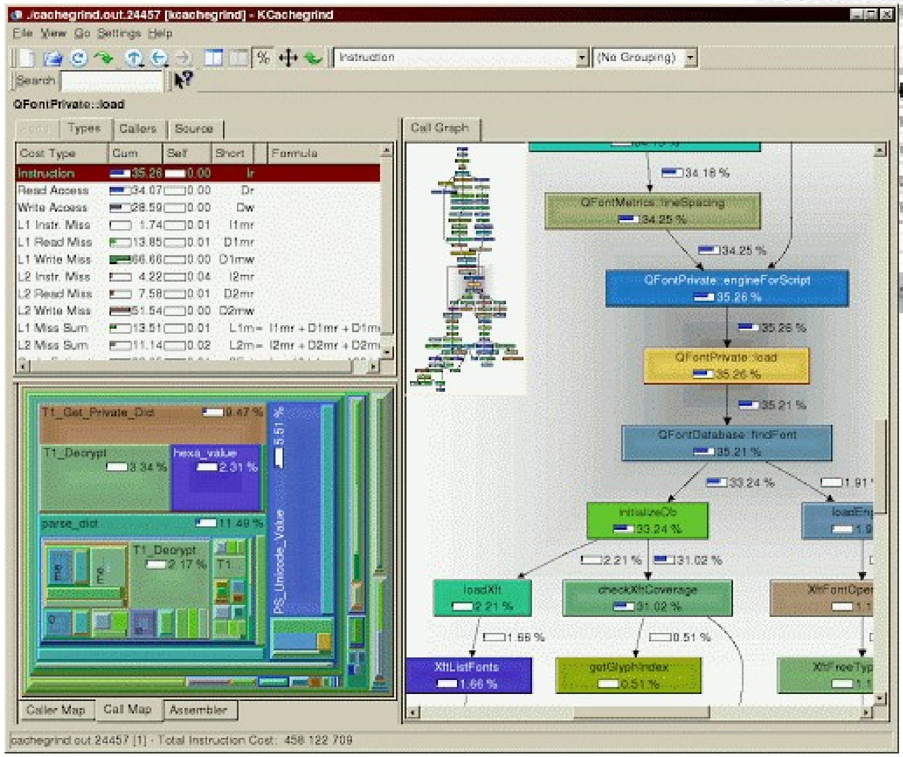
- Use guesstimates for the first job
- Then analyze the accounting information
- Extrapolate for next jobs



```
[root@mbackMGT ~]# sacct -o JobID,MaxRSS,ReqMem,minCPU,AllocCPU,MaxDiskWrite
       JobID     MaxRSS     ReqMem     MinCPU  AllocCPUS MaxDiskWrite
------------ ---------- ---------- ---------- ---------- ------------
1309590                     2500Mc                    48
1309590.bat+      6904K     2500Mc   00:00:00         48        0.08M
1309590.0     28372424K     2500Mc 98-01:18:+         48     4443125M
1312190                     8000Mc                    32
1312190.bat+      8284K     8000Mc   00:00:00         32           2M
1312190.0       249.61G     8000Mc 25-12:51:+         32        0.02M
1313223                     8000Mc                    32
1313223.bat+      8232K     8000Mc   00:00:00         32           2M
1313223.0    262164304K     8000Mc 25-15:56:+         32        0.02M
1313732                     2000Mc                     1
1313732.bat+    277028K     2000Mc 4-02:57:25          1          21M
1313733                     2000Mc                     1
1313733.bat+    324436K     2000Mc 4-02:58:51          1          21M
1313786                     2000Mc                     1
1313786.bat+    303100K     2000Mc 4-02:55:40          1          21M
1313787                     2000Mc                     1
1313787.bat+    350368K     2000Mc 4-02:55:15          1          21M
1313860                     2000Mc                     1
1313860.bat+    332972K     2000Mc 4-02:54:19          1          21M
1313861                     2000Mc                     1
1313861.bat+    380640K     2000Mc 4-02:54:11          1          21M
1355314                     1992Mc                     8
1355314.bat+  10193084K     1992Mc 1-01:36:18          8       81990M
1357875                     1992Mc                     8
```

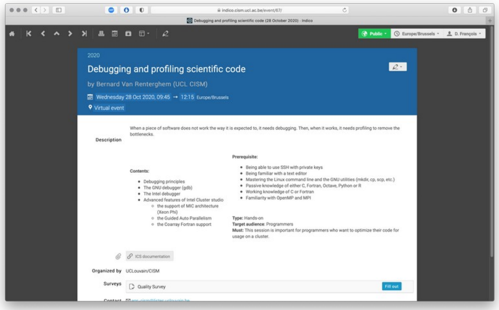
Best approach

Use profiling tools...


./cachegrind.out.24457 [kcachegrind] - KCachegrind
File View Go Settings Help
Instruction
(No Grouping)
Search
QFontPrivate::load
Types Callers Source
Call Graph
Caller Map Call Map Assembler

Cost Type	Cum	Self	Short	Formula
Instruction	35.26	0.00	Ir	
Read Access	34.07	0.00	Dr	
Write Access	28.59	0.00	Dw	
L1 Instr. Miss	1.74	0.01	I1mr	
L1 Read Miss	13.85	0.01	D1mr	
L1 Write Miss	66.66	0.00	D1mw	
L2 Instr. Miss	4.22	0.04	I2mr	
L2 Read Miss	7.58	0.01	D2mr	
L2 Write Miss	51.54	0.00	D2mw	
L1 Miss Sum	13.51	0.01	L1m=	I1mr + D1mr + D1mw
L2 Miss Sum	11.14	0.02	L2m=	I2mr + D2mr + D2mw

cachegrind.out.24457 [1] - Total Instruction Cost: 458 122 709


indico.cism.ucl.ac.be/event/67/

Debugging and profiling scientific code (28 October 2020) · Indico

Public · Europe/Brussels · D. François

2020

Debugging and profiling scientific code

by Bernard Van Renterghem (UCL CISM)

Wednesday 28 Oct 2020, 09:45 → 12:15 Europe/Brussels

Virtual event

Description: When a piece of software does not work the way it is expected to, it needs debugging. Then, when it works, it needs profiling to remove the bottlenecks.

Contents:

- Debugging principles
- The GNU debugger (gdb)
- The Intel debugger
- Advanced features of Intel Cluster studio
  - the support of MIC architecture (Xeon Phi)
  - the Guided Auto Parallelism
  - the Coarray Fortran support

Prerequisite:

- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Passive knowledge of either C, Fortran, Octave, Python or R
- Working knowledge of C or Fortran
- Familiarity with OpenMP and MPI

Type: Hands-on
Target audience: Programmers
Must: This session is important for programmers who want to optimize their code for usage on a cluster.

ICS documentation

Organized by: UCLouvain/CISM

Surveys: Quality Survey — Fill out

You will learn how to:

Create a parallel job
Request distributed resources

with

slurm
workload manager

You will learn how to:

Create a parallel job
Request distributed resources

6 typical use cases:

1. MPI programs
2. Multithreaded programs
3. Master/slave
4. Embarrassingly parallel
5. Heterogeneous jobs
6. Accelerators

Use case 1: Message passing

You have a program *myprog* that uses an MPI library

e.g. OpenMPI, Intel MPI, MVAPICH, etc.

You want	You ask
N CPUs, to launch *N* MPI processes	--ntasks=*N*
You use	srun ./myprog (Intel MPI and OpenMPI >= 1.5) mpirun ./myprog (OpenMP<1.5 & mvapich)

submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=8

module load OpenMPI

srun ./myprog
```

Use case 1: Message passing

You want	You ask
N CPUs	--ntasks=*N*
N CPUs spread across distinct nodes	--ntasks=*N* --nodes=*N* *or* *--ntasks=N --ntasks-per-node=1*
N CPUs spread across distinct nodes and nobody else around	--ntasks=*N* --nodes=*N* --exclusive
N CPUs spread across *N/2* nodes	--ntasks=*N* --ntasks-per-node=2
N CPUs on the same node	--ntasks=*N* --ntasks-per-node=*N* *or* *--ntasks=N --nodes=1*

Use case 2: Multithreading

You have a program *myprog* that spawns several threads/processes

e.g. OpenMP, PThreads, TBB, parallel libraries like OpenBLAS, Python multiprocessing, etc.

You want	You ask
N CPUs to launch *N* processes or threads on the same node	--cpus-per-task=*N*
You use	OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK export OMP_NUM_THREADS MKL_NUM_THREADS=$SLURM_CPUS_PER_TASK export MKL_NUM_THREADS etc. srun ./myprog

submit.sh

```
#! /bin/bash
#
#SBATCH --cpus-per-task=8

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun ./myprog
```

Use case 2: Multithreading

You have a program *myprog* that spawns several threads/processes

e.g. OpenMP, PThreads, TBB, parallel libraries like OpenBLAS, Python multiprocessing, etc.

You want	You ask
All the CPUs on the node	--exclusive --mem=0
You use	

submit.sh

```
#! /bin/bash
#
#SBATCH --exclusive
#SBATCH --mem=0

srun ./myprog
```

Use case 3: Master/Slave

You have a program *master* that coordinates several *slave* programs

e.g. Matlab with Multicore,

You want	You ask
N CPUs to launch *N* processes or threads on the same node	--ntasks=*N* --ntasks-per-node=*N*
You use	file multi.conf srun --multi-prog multi.conf

submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=8

srun --multi-prog multi.conf
```

multi.conf

```
# multi.conf for --multi-prog
0: ./master
1-7: ./slave
```

Use case 4: Embarrassingly parallel

You have a program *myprog* of which several instances must run

e.g. to process **distinct parameters values**, distinct files, etc.

You want	You ask
N CPUs to launch *N* completely independent jobs	--array=1-*N*
You use	$SLURM_TASK_ARRAY_ID srun ./myprog

submit.sh

```
#! /bin/bash
#
#SBATCH --array=1-8

srun ./myprog $SLURM_TASK_ARRAY_ID
```

Use case 4: Embarrassingly parallel

You have a program *myprog* of which several instances must run

e.g. to process distinct parameters values, **distinct files**, etc.

You want	You ask
N CPUs to launch *N* completely independent jobs	--array=1-*N*
You use	$SLURM_TASK_ARRAY_ID srun ./myprog

submit.sh

```
#! /bin/bash
#
#SBATCH –array=0-7 # assuming 8 files

FILES=(/path/to/data/*)

srun ./myprog ${FILES[$SLURM_TASK_ARRAY_ID]}
```

Use case 5: Heterogeneous jobs

You have non-homogeneous requests

(e.g. 1cpu 4GB + 10cpu 1G, or 10 nodes + 1 GPU)

submit.sh

```
#! /bin/bash

#SBATCH --cpus-per-task=1 --mem-per-cpu=1g
--ntasks=4
#SBATCH packjob
#SBATCH --cpus-per-task=4 --mem-per-cpu=16g
--ntasks=1

echo Step 1
srun -l --pack-group=1 hostname
sleep 3

echo Step 2
srun -l bash -c "hostname;ulimit -m" :\
        bash -c "hostname;ulimit -m"
sleep 3

scancel $SLURM_JOB_ID+1

echo Step 3
srun -l bash -c "hostname;ulimit -m"
```

Use case 5: Heterogeneous jobs

You have non-homogeneous requests

submit.sh

```
$ cat res
Step 1
P1 0: lm3-w005.cluster
Step 2
4: lm3-w005.cluster
4: 67108864
1: lm3-w013.cluster
1: 4194304
2: lm3-w013.cluster
2: 4194304
3: lm3-w013.cluster
3: 4194304
0: lm3-w013.cluster
0: 4194304
Step 3
P0 1: lm3-w013.cluster
P0 1: 4194304
P0 3: lm3-w013.cluster
P0 2: lm3-w013.cluster
P0 2: 4194304
P0 0: lm3-w013.cluster
P0 3: 4194304
P0 0: 4194304
```

Use case 6: Accelerators (GPUs)

You want to use GPUs

You want	You ask
N GPUs	--gpus=N (Slurm 19.05 and newer) --gres=gpu:N (older Slurm versions)
1 specific GPU (e.g. TeslaV100)	--gpus=TeslaV100:1 (Slurm 19.05 and newer) --gres=gpu:TeslaV100:1 (older Slurm versions)

submit.sh

```
#! /bin/bash

#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=1g
#SBATCH --gres=gpu:1

module load CUDA # or cuda on some clusters
nvidia-smi
```

Hybrid jobs

with for instance MPI and OpenMP

submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=8
#SBATCH --ncpus-per-task=4

module load OpenMPI
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun ./myprog
```

or even a job array of hybrid jobs...

submit.sh

```
#! /bin/bash
#
#SBATCH --array=1-10
#SBATCH --ntasks=8
#SBATCH --ncpus-per-task=4

module load OpenMPI
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun ./myprog $SLURM_TASK_ARRAY_ID
```

Scripting submissions

Only if few jobs and complex arguments

otherwise use job arrays

Step 1: use command line options to sbatch rather than submission script. For instance,

submit.sh

```
#! /bin/bash
#
#SBATCH --ncpus-per-task=4

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun ./myprog
```

becomes

```
$ export OMP_NUM_THREADS=4
$ sbatch --ntasks=8 --ncpus-per-task=4 --wrap "srun ./myprog"
```

Scripting submissions

Only if few jobs and complex arguments

otherwise use job arrays

Step 2: use tips from session 'Parallel Computing'

e.g. you have several files **data_red.csv**, **data_blue.csv**, **data_green.csv** and *myprog* takes the file in argument

```
$ ls data*csv | xargs -n1 -I{} sbatch ... --wrap “./myprog {}”
```

will be equivalent to

```
$ sbatch ... --wrap “./myprog data_red.csv”
$ sbatch ... --wrap “./myprog data_blue.csv”
$ sbatch ... --wrap “./myprog data_green.csv”
```

Scripting submissions

Only if few jobs and complex arguments

otherwise use job arrays

use tips from session 'Parallel Computing'

e.g. you have *myprog* parameter one ranging from 1 to 3 and parameter two ranging from A to C

```
$ parallel sbatch ... --wrap \"./myprog {1} {2}\" ::: {1..3} ::: {A..C}
```

will be equivalent to

```
$ sbatch ... --wrap "./myprog 1 A"
$ sbatch ... --wrap "./myprog 1 B"
$ sbatch ... --wrap "./myprog 1 C"
$ sbatch ... --wrap "./myprog 2 A"
$ sbatch ... --wrap "./myprog 2 B"
...
```

Packing jobs

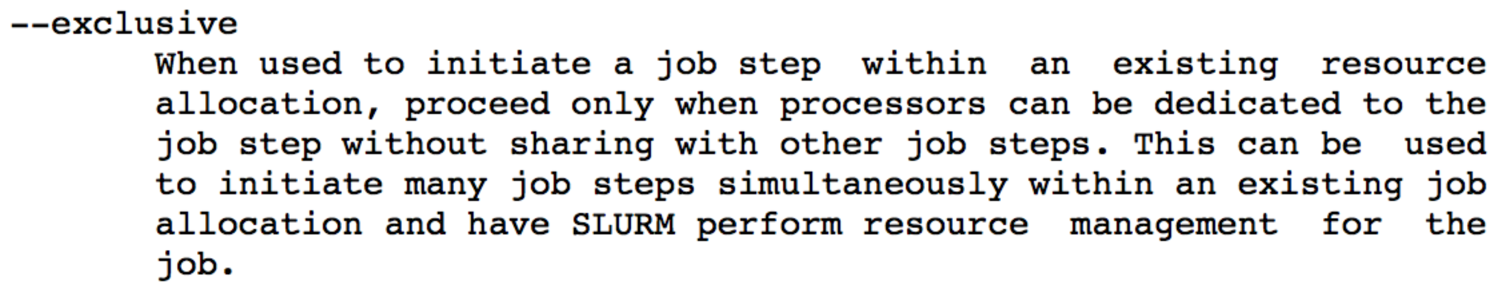
when each step lasts less than ~30 mins

to avoid spending as much time handling jobs as running them

e.g. your program *myprog* lasts one minute but need to be run with argument from 1 to 1000

submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=8

for i in {1..1000}
do
    srun -n1 --exclusive ./myprog $i &
done
wait
```



```
--exclusive
       When used to initiate a job step  within  an  existing  resource
       allocation, proceed only when processors can be dedicated to the
       job step without sharing with other job steps. This can be  used
       to initiate many job steps simultaneously within an existing job
       allocation and have SLURM perform resource  management  for  the
       job.
```

Packing jobs

when each step lasts less than ~30 mins

to avoid spending as much time handling jobs as running them

You can also use **xargs** or **parallel** inside your submission script:

submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=8

parallel -P 8 srun  -n1 --exclusive ./myprog ::: {1..1000}
```

Packing jobs

when each step lasts less than ~30 mins

to avoid spending as much time handling jobs as running them

You can also use **xargs** or **parallel** inside your submission script:

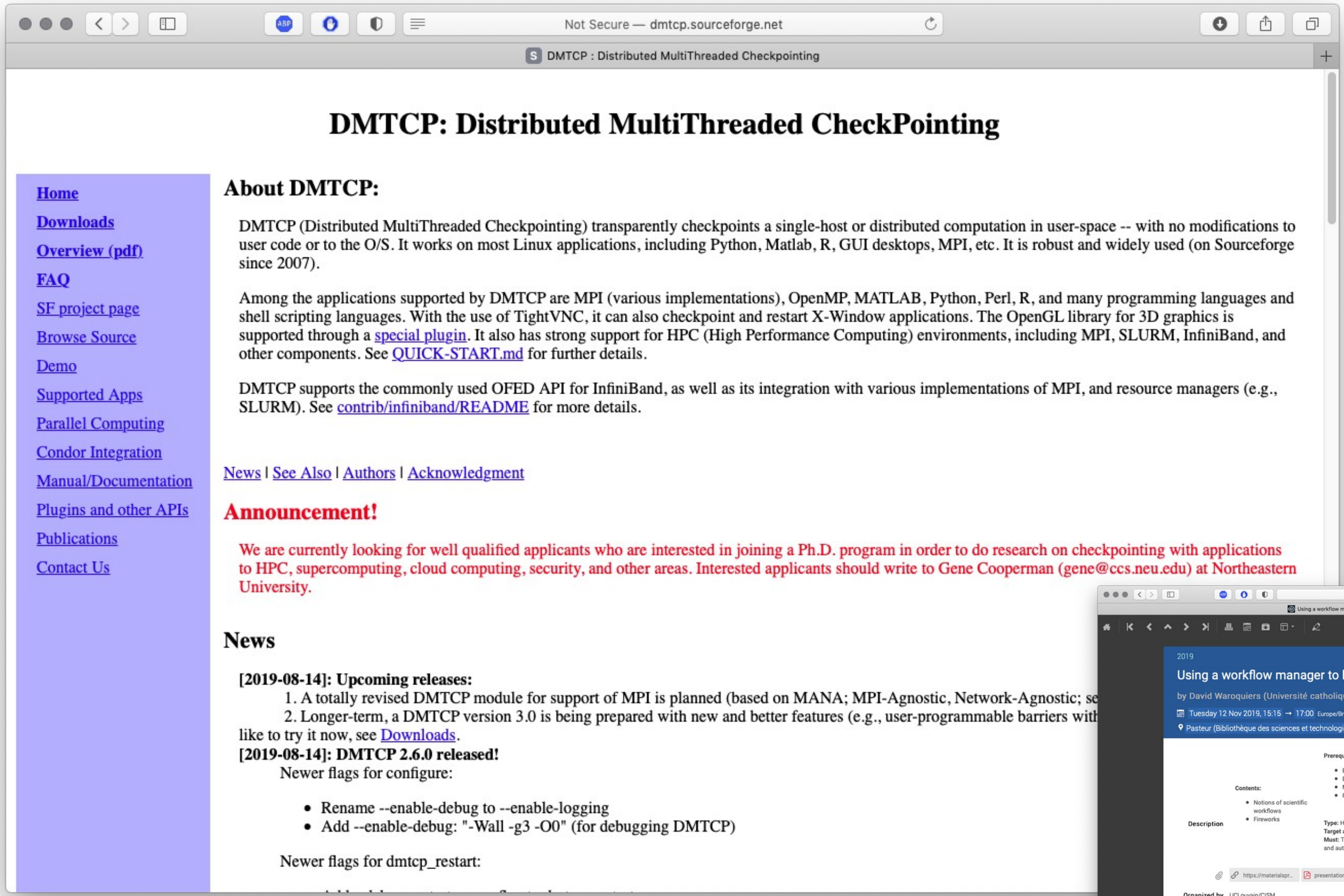
submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=8

ls data* | xargs -n1 -P 8 srun -n1 --exclusive ./myprog
```

Checkpointing

when your jobs are toooooo loooooong

compared with the cluster maximum walltimes



Not Secure — dmtcp.sourceforge.net
DMTCP : Distributed MultiThreaded Checkpointing

DMTCP: Distributed MultiThreaded CheckPointing

Home
Downloads
Overview (pdf)
FAQ
SF project page
Browse Source
Demo
Supported Apps
Parallel Computing
Condor Integration
Manual/Documentation
Plugins and other APIs
Publications
Contact Us

About DMTCP:

DMTCP (Distributed MultiThreaded Checkpointing) transparently checkpoints a single-host or distributed computation in user-space -- with no modifications to user code or to the O/S. It works on most Linux applications, including Python, Matlab, R, GUI desktops, MPI, etc. It is robust and widely used (on Sourceforge since 2007).

Among the applications supported by DMTCP are MPI (various implementations), OpenMP, MATLAB, Python, Perl, R, and many programming languages and shell scripting languages. With the use of TightVNC, it can also checkpoint and restart X-Window applications. The OpenGL library for 3D graphics is supported through a special plugin. It also has strong support for HPC (High Performance Computing) environments, including MPI, SLURM, InfiniBand, and other components. See QUICK-START.md for further details.

DMTCP supports the commonly used OFED API for InfiniBand, as well as its integration with various implementations of MPI, and resource managers (e.g., SLURM). See contrib/infiniband/README for more details.

News | See Also | Authors | Acknowledgment

Announcement!

We are currently looking for well qualified applicants who are interested in joining a Ph.D. program in order to do research on checkpointing with applications to HPC, supercomputing, cloud computing, security, and other areas. Interested applicants should write to Gene Cooperman (gene@ccs.neu.edu) at Northeastern University.

News

[2019-08-14]: Upcoming releases:
1. A totally revised DMTCP module for support of MPI is planned (based on MANA; MPI-Agnostic, Network-Agnostic; se
2. Longer-term, a DMTCP version 3.0 is being prepared with new and better features (e.g., user-programmable barriers with

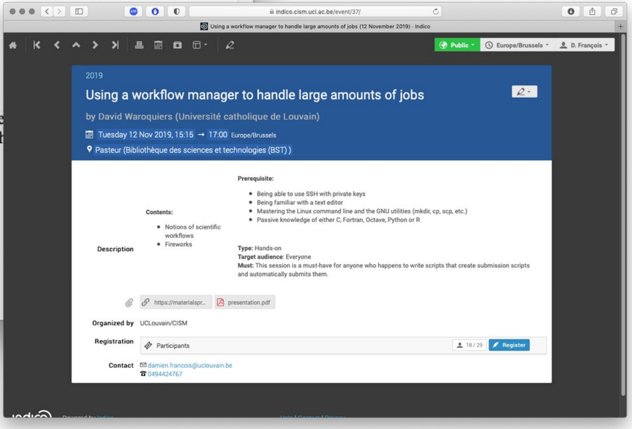
like to try it now, see Downloads.

[2019-08-14]: DMTCP 2.6.0 released!

Newer flags for configure:

- Rename --enable-debug to --enable-logging
- Add --enable-debug: "-Wall -g3 -O0" (for debugging DMTCP)

Newer flags for dmtcp_restart:



indico.cism.ucl.ac.be/event/37/
Using a workflow manager to handle large amounts of jobs (12 November 2019) · Indico
Public
Europe/Brussels
D. François

2019

Using a workflow manager to handle large amounts of jobs

by David Waroquiers (Université catholique de Louvain)

Tuesday 12 Nov 2019, 15:15 → 17:00 Europe/Brussels

Pasteur (Bibliothèque des sciences et technologies (BST))

Description

Contents:
- Notions of scientific workflows
- Fireworks

Prerequisite:
- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Passive knowledge of either C, Fortran, Octave, Python or R

Type: Hands-on
Target audience: Everyone
Must: This session is a must-have for anyone who happens to write scripts that create submission scripts and automatically submits them.

https://materialspr... presentation.pdf

Organized by UCLouvain/CISM

Registration Participants 18 / 29 Register

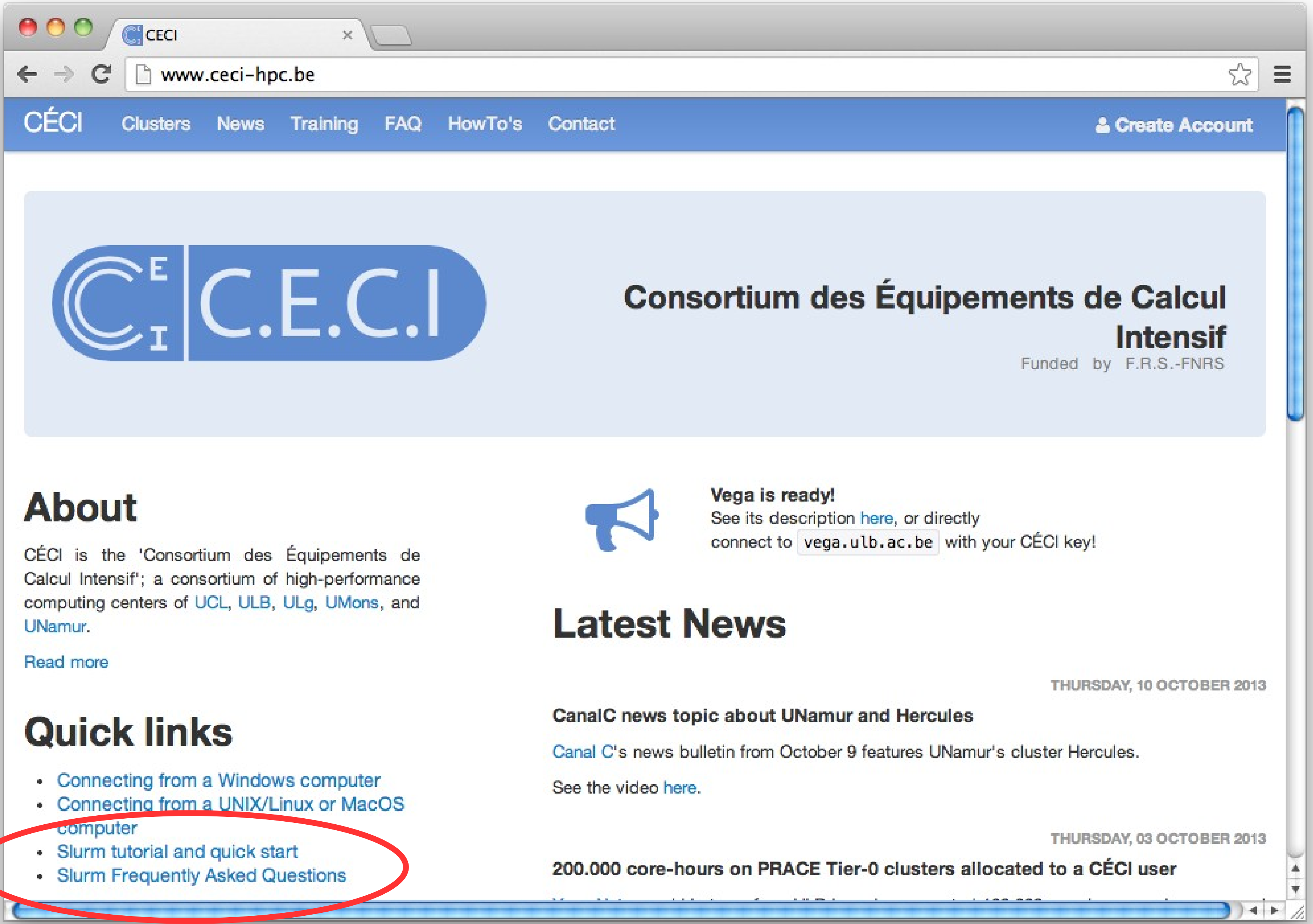
Contact damien.francois@uclouvain.be
0494424767

Summary

- Choose number of **processes: --ntasks**
- Choose number of **threads: --cpu-per-task**

- Launch processes with srun or mpirun
- Set multithreading with OMP_NUM_THREADS

- You can use $SLURM_PROC_ID
  $SLURM_TASK_ID
  $SLURM_TASK_ARRAY_ID



Consortium des Équipements de Calcul Intensif
Funded by F.R.S.-FNRS

About

CÉCI is the 'Consortium des Équipements de Calcul Intensif'; a consortium of high-performance computing centers of UCL, ULB, ULg, UMons, and UNamur.

Read more

Quick links

- Connecting from a Windows computer
- Connecting from a UNIX/Linux or MacOS computer
- Slurm tutorial and quick start
- Slurm Frequently Asked Questions

Vega is ready!
See its description here, or directly connect to `vega.ulb.ac.be` with your CÉCI key!

Latest News

THURSDAY, 10 OCTOBER 2013

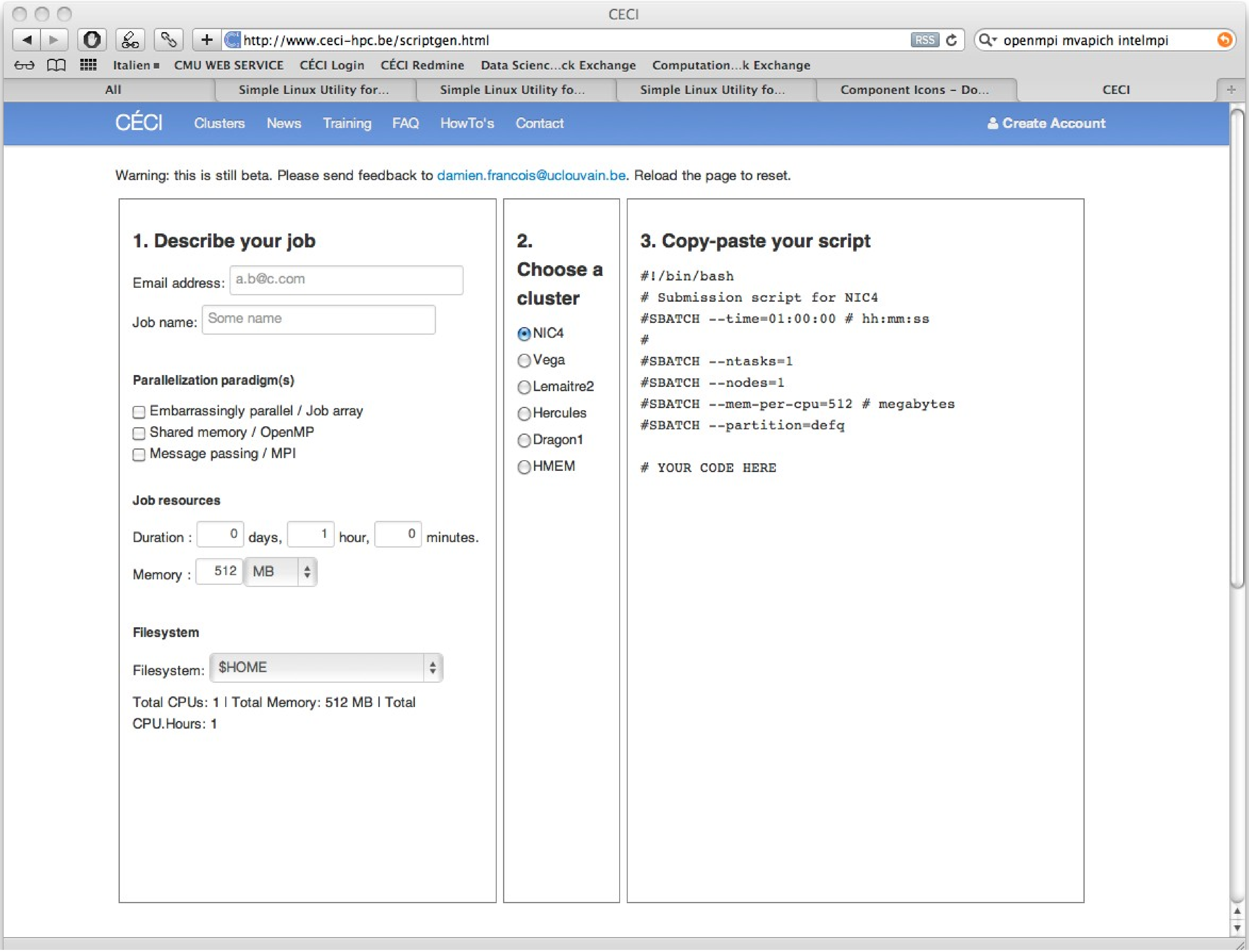
CanalC news topic about UNamur and Hercules

Canal C's news bulletin from October 9 features UNamur's cluster Hercules.

See the video here.

THURSDAY, 03 OCTOBER 2013

200.000 core-hours on PRACE Tier-0 clusters allocated to a CÉCI user



Warning: this is still beta. Please send feedback to damien.francois@uclouvain.be. Reload the page to reset.

1. Describe your job

Email address: a.b@c.com

Job name: Some name

Parallelization paradigm(s)

☐ Embarrassingly parallel / Job array
☐ Shared memory / OpenMP
☐ Message passing / MPI

Job resources

Duration : 0 days, 1 hour, 0 minutes.

Memory : 512 MB

Filesystem

Filesystem: $HOME

Total CPUs: 1 | Total Memory: 512 MB | Total CPU.Hours: 1

2. Choose a cluster

[✓ NIC4]
Vega
Lemaitre2
Hercules
Dragon1
HMEM

3. Copy-paste your script

```
#!/bin/bash
# Submission script for NIC4
#SBATCH --time=01:00:00 # hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=512 # megabytes
#SBATCH --partition=defq

# YOUR CODE HERE
```