slurm

workload manager

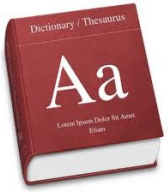
Until now:

- 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.

What is a job scheduler?

Job scheduler

From Wikipedia, the free encyclopedia

This article is about a class of software. For the mathematical problem in Computer Science, see Job Shop Scheduling. For other uses, see Scheduling (disambiguation).

A **job scheduler** is a computer application for controlling unattended background program execution (commonly called batch processing).[1]

Synonyms are **batch system**, **Distributed Resource Management System** (DRMS), and **Distributed Resource Manager** (DRM). Today's job schedulers, often termed **workload automation**, typically provide a graphical user interface and a single point of control for definition and monitoring of background executions in a distributed network of computers. Increasingly, job schedulers are required to orchestrate the integration of real-time business activities with traditional background IT processing across different operating system platforms and business application environments.

Job scheduling should not be confused with process scheduling, which is the assignment of currently running processes to CPUs by the operating system.

Job scheduler/Resource manager :

Piece of software which:

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

Two computers are available for 10h

You go, then you go. You wait.

and sets up the environment
for parallel and distributed computing

Resources:

CPU cores

Memory

Disk space

Network

Accelerators

Software

Licenses

Lawrence Livermore National Laboratory

Motto	"Science and Technology in the National Interest"
Established	1952 by the University of California
Research type	Nuclear and basic science
Budget	US$1.5 billion
Director	Penrose (Parney) Albright
Staff	6,800
Location	Livermore, California
Campus	320 hectares (790 acres)
Operating agency	Lawrence Livermore National Security, LLC
Website	www.llnl.gov www.llnsllc.com

Slurm Workload Manager

Slurm is an open-source workload manager designed for Linux clusters of all sizes. It provides three key functions. First it allocates exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (typically a parallel job) on a set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work.

While other workload managers do exist, Slurm is unique in several respects:

- **Scalability**: It is designed to operate in a heterogeneous cluster with up to tens of millions of processors.
- **Performance**: It can accept 1,000 job submissions per second and fully execute 500 simple jobs per second (depending upon hardware and system configuration).
- **Free and Open Source**: Its source code is freely available under the GNU General Public License.
- **Portability**: Written in C with a GNU autoconf configuration engine. While initially written for Linux, Slurm has been ported to a diverse assortment of systems.
- **Power Management**: Job can specify their desired CPU frequency and power use by job is recorded. Idle resources can be powered down until needed.
- **Fault Tolerant**: It is highly tolerant of system failures, including failure of the node executing its control functions.
- **Flexibility**: A plugin mechanism exists to support various interconnects, authentication mechanisms, schedulers, etc. These plugins are documented and simple enough for the motivated end user to understand the source and add functionality.
- **Resizable Jobs**: Jobs can grow and shrink on demand. Job submissions can specify size and time limit ranges.
- **Status Jobs**: Status running jobs at the level of individual tasks to help identify load imbalances and other anomalies.

Slurm provides workload management on many of the most powerful computers in the world. On the June 2013 Top500 list, five of the ten top systems use Slurm including the number one system. These five systems alone contain over 5.7 million cores. A few of the systems using Slurm are listed below:

- Tianhe-2 designed by The National University of Defense Technology (NUDT) in China has 16,000 nodes, each with two Intel Xeon IvyBridge processors and three Xeon Phi processors for a total of 3.1 million cores and a peak performance of 33.86 Petaflops.

Slurm

Free and open-source

Mature (exists since ~2003)

Very active community

Many success stories

Runs 60% of TOP500 systems

Also an intergalactic soft drink

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	361,760	19,590.0	25,326.3	2,272
4	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
5	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
6	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
7	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
8	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6		
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9		

TOP 500 The List.

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	361,760	19,590.0	25,326.3	2,272
4	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
5	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
6	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
7	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,719
8	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
9	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6		
10	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9		

TOP 500 The List.

You will learn how to:

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

with

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

```
#!/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
```

Regular Bash comment

Regular Bash commands

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	--email-type= BEGIN|END|FAILED|ALL| TIME_LIMIT_90 --email-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

Constraints and resources

You want	You ask
To choose a specific feature (e.g. a processor type or a NIC type)	--constraint
To use a specific resources (e.g. a gpu)	--gres
To reserve a whole node for yourself	--exclusive
To chose a partition	--partition

```
[dfr@manneback ~]$ sinfo -o "%40N %.6D %.5P %.4c %.7m %30f %50G"
NODELIST                                 NODES PARTI CPUS  MEMORY FEATURES                       GRES
mb-wes[251-252]                              2  Def*   12   96734 Westmere,Xeon,X5675            localscratch:1650,gpu:TeslaC2050/C2075:1
mb-har[001-009,011-014]                     13  Def*    8   15947 Harpertown,Xeon,L5420          localscratch:166
mb-har[101,103-115]                         14  Def*    8   32108 Harpertown,Xeon,L5420          localscratch:73
mb-har102                                    1  Def*    8   32108 Harpertown,Xeon,L5420          localscratch:146
mb-har[121-140]                             20  Def*    8  13930+ Harpertown,Xeon,X5460          localscratch:20
mb-ivy[201-208]                              8  Def*   16   64386 IvyBridge,Xeon,E5-2650v2       localscratch:172
mb-neh070                                    1  Def*    8   24019 Nehalem,Xeon,X5550             localscratch:814,gpu:TeslaC1060/M1060:2
mb-neh[201-204]                              4  Def*    8   24020 Nehalem,Xeon,L5520             localscratch:916
mb-neh[205-212]                              8  Def*    8   24020 Nehalem,Xeon,L5520             localscratch:814
mb-opt[111-116]                              6  Def*   32 112898+ Bulldozer,Opteron,6276         localscratch:355
mb-sab040                                    1  Def*   16   64398 SandyBridge,Xeon,E5-2660       localscratch:458,gpu:TeslaM2090:2,mic:5110P:1
mb-sab103                                    1  Def*   32  129013 SandyBridge,Xeon,E5-4620       localscratch:3564
mb-sab[001-021]                             21  Def*   16  64258+ SandyBridge,Xeon,E5-2650       localscratch:355
mb-sab[101-102]                              2  Def*   32  258285 SandyBridge,Xeon,E5-4640       localscratch:355
mb-has[001-026]                             26   Zoe   16   64171 Haswell,Xeon,E5-2630v3         localscratch:447
mb-har[021,023,025,027,029,031,034-035,0    16   cp3    8   15947 Harpertown,Xeon,E5420          localscratch:46
mb-ivy[211-217,219-227]                     16   cp3   48  129022 IvyBridge,Xeon,E5-2695v2       localscratch:814
mb-har[022,024,026,028,030,032-033,036,0    15   cp3    8   15947 Harpertown,Xeon,L5420          localscratch:46
mb-opt[011-042,055-064,066-079]             56   cp3   16  32103+ K10,Opteron,6134               localscratch:126
mb-opt[043-048]                              6   cp3   16   64423 K10,Opteron,6134               localscratch:814
mb-opt[049-054]                              6   cp3   16  32103+ K10,Opteron,6134               localscratch:916
mb-sab[081-084,086-090]                      9   cp3   32   64386 SandyBridge,Xeon,E5-2670       localscratch:355
```

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

We will use the *stress* program to simulate a real program

Real programs

- use CPUs,
- consume memory, and
- run for some time

We will use the *stress* program to simulate a real program

the *stress* program

- uses CPUs,
- consumes memory, and
- runs for some time (seconds)

```
stress --timeout 300 --vm 1 --vm-bytes 128M --cpu 2
```

module load stress

We will use the *stress* program to simulate a real program

the *stress* program

- uses CPUs,
- consumes memory, and
- runs for some time

```
stress --timeout 300 --vm 1 --vm-bytes 128M --cpu 2
```

What parameters should you use in Slurm to run the above program successfully?

Try and submit a job (name it 'stress')

We will use the *stress* program to simulate a real program

the *stress* program

- uses CPUs,
- consumes memory, and
- runs for some time

```
stress --timeout 300 --vm 1 --vm-bytes 128M --cpu 2
```

What happens if you underestimate those parameters in your Slurm submission script?

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:~
```

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

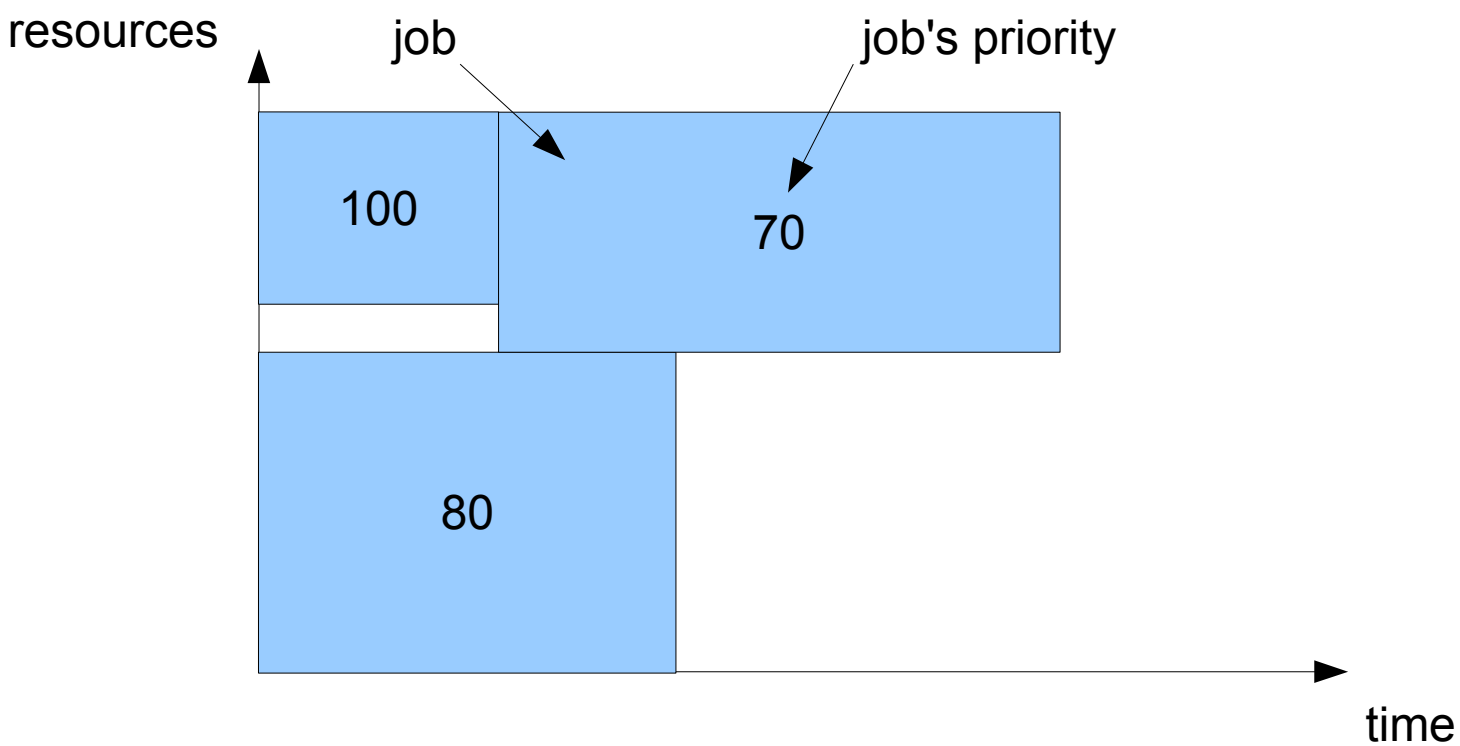
```
Job_priority =
        (PriorityWeightAge) * (age_factor) +
        (PriorityWeightFairshare) * (fair-share_factor) +
        (PriorityWeightJobSize) * (job_size_factor) +
        (PriorityWeightPartition) * (partition_factor) +
        (PriorityWeightQOS) * (QOS_factor) +
        SUM(TRES_weight_cpu * 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

A word about backfill

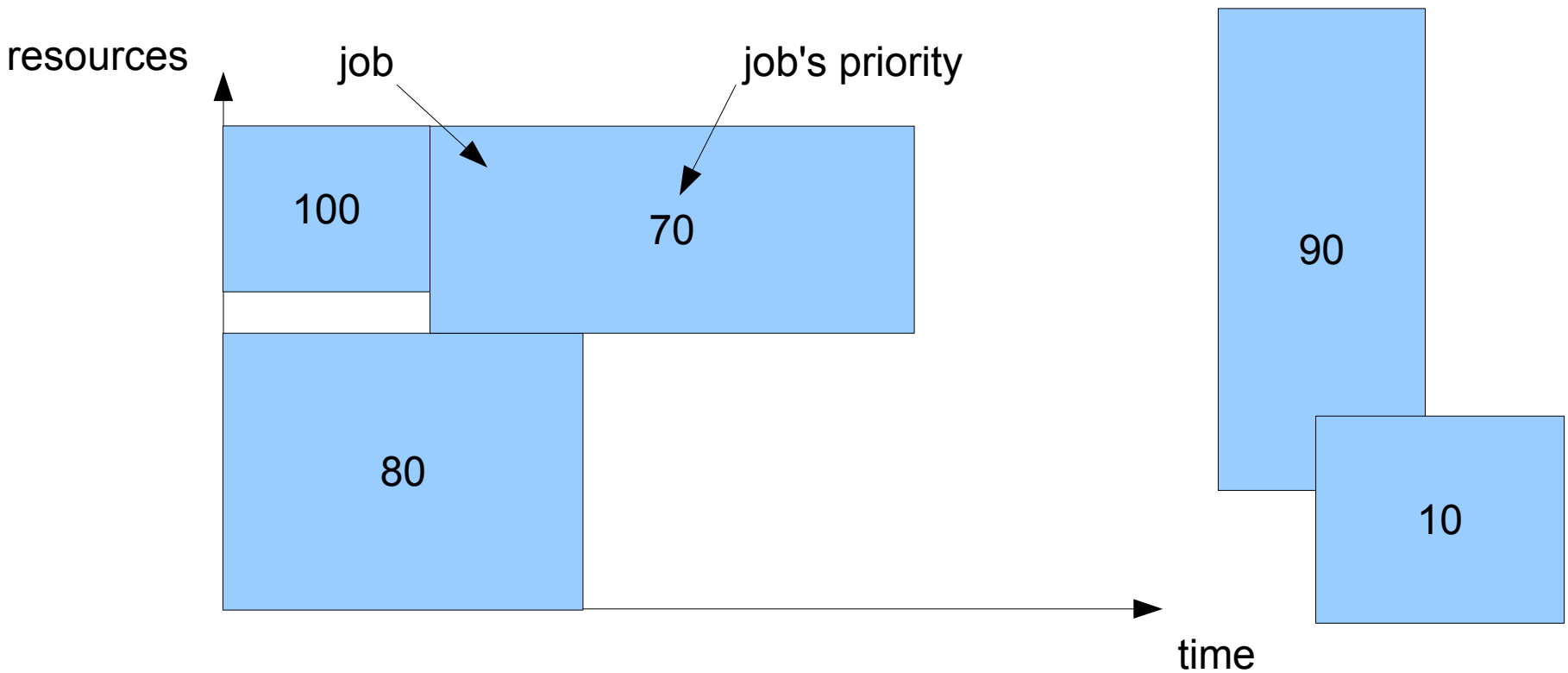
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



A job is a number of cpus times duration

A word about backfill

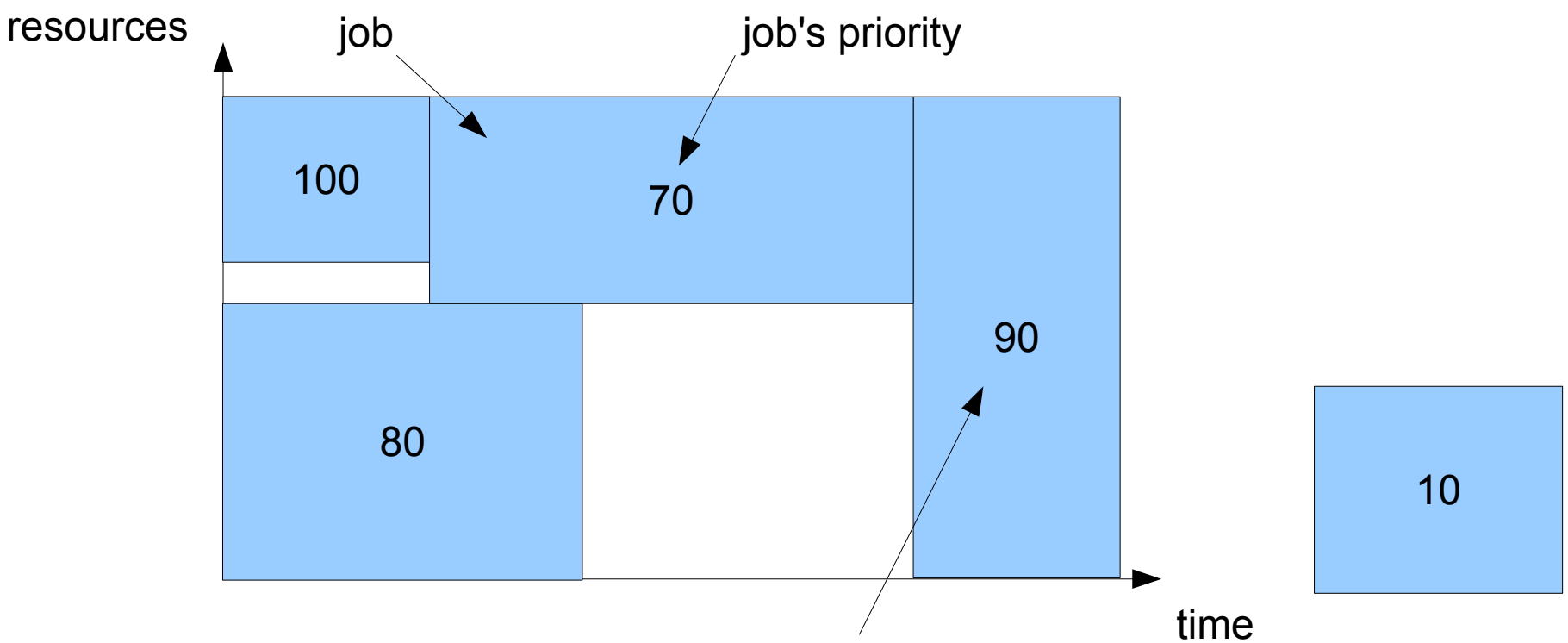
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



Two more jobs to schedule

A word about backfill

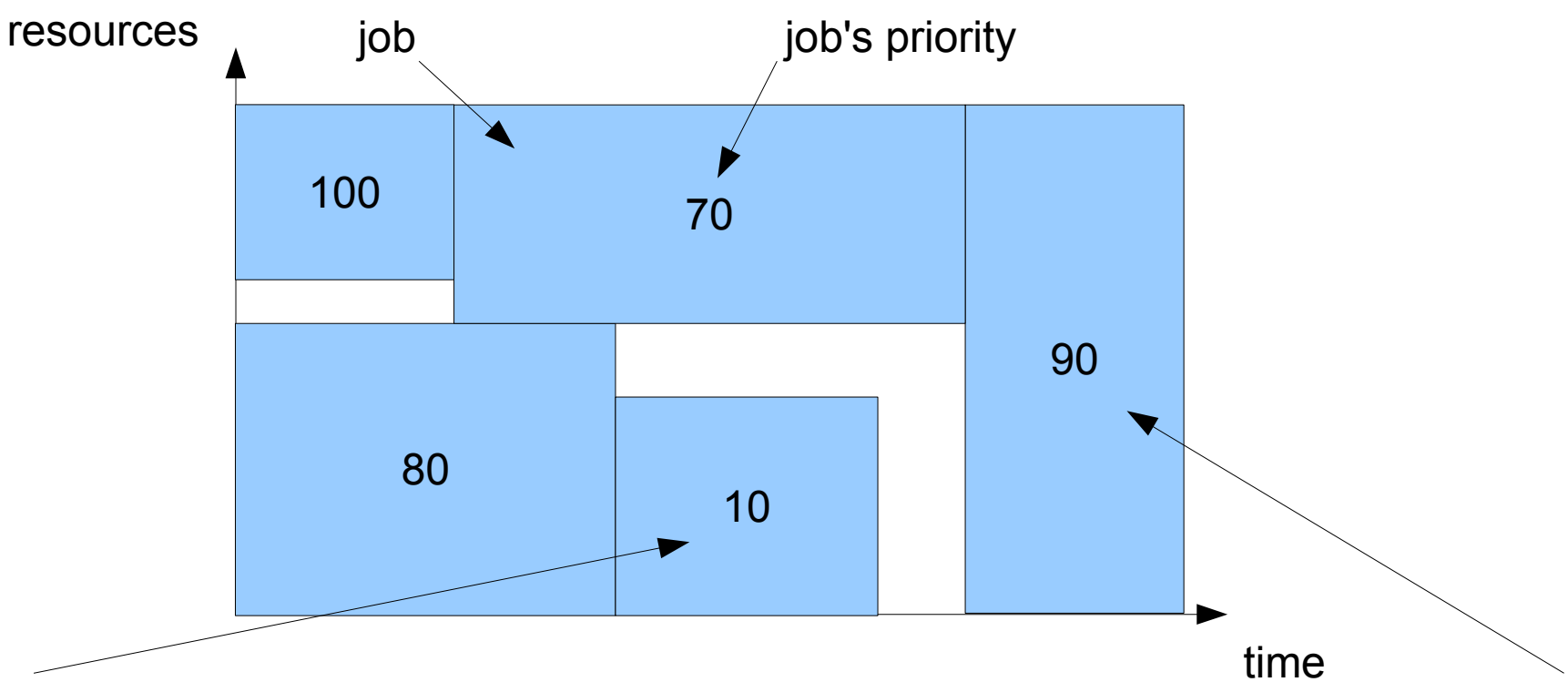
The rule: a job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



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

4. Monitor your job

- squeue
- sprio
- sstat
- sview

```
SSTAT(1)                                Slurm components
                    SSTAT(1)

NAME
       sstat  - Display various status informa-
       tion of a running job/step.

SYNOPSIS
       sstat [OPTIONS...]

DESCRIPTION
       Status  information  for  running   jobs
       invoked with SLURM.

       The  sstat  command  displays job status
       information  for  your  analysis.    The
       sstat  command displays information per-
       taining to CPU, Task, Node, Resident Set
       Size (RSS) and Virtual Memory (VM).  You
       can tailor the output with  the  use  of
:
```

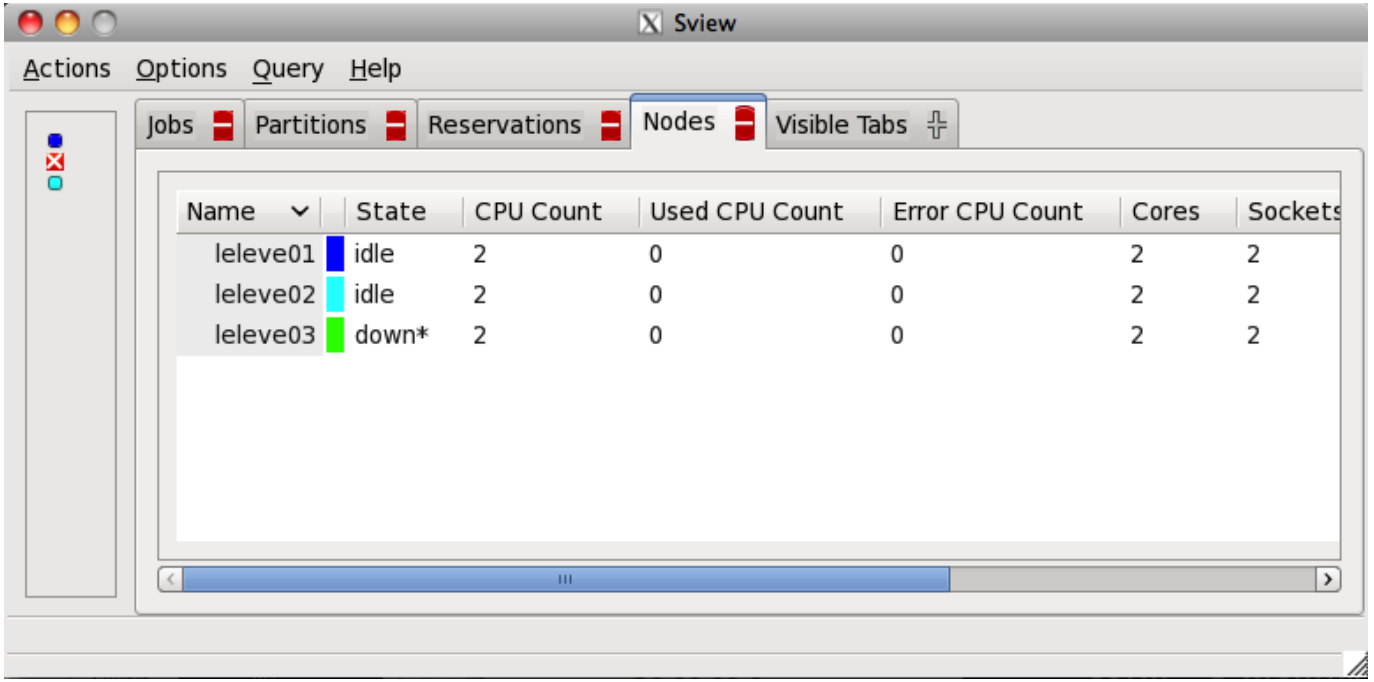
4. Monitor your job

- squeue
- sprio
- sstat
- sview

```
dfr@hmem00:~ # sstat -j jobid
dfr@hmem00:~ # sstat -j jobid --pidformat
dfr@hmem00:~ # sstat -o AveCPU,AveRSS,...
dfr@hmem00:~ #
```

4. Monitor your job

- squeue
- sprio
- sstat
- sview



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

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 -j 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
:
```

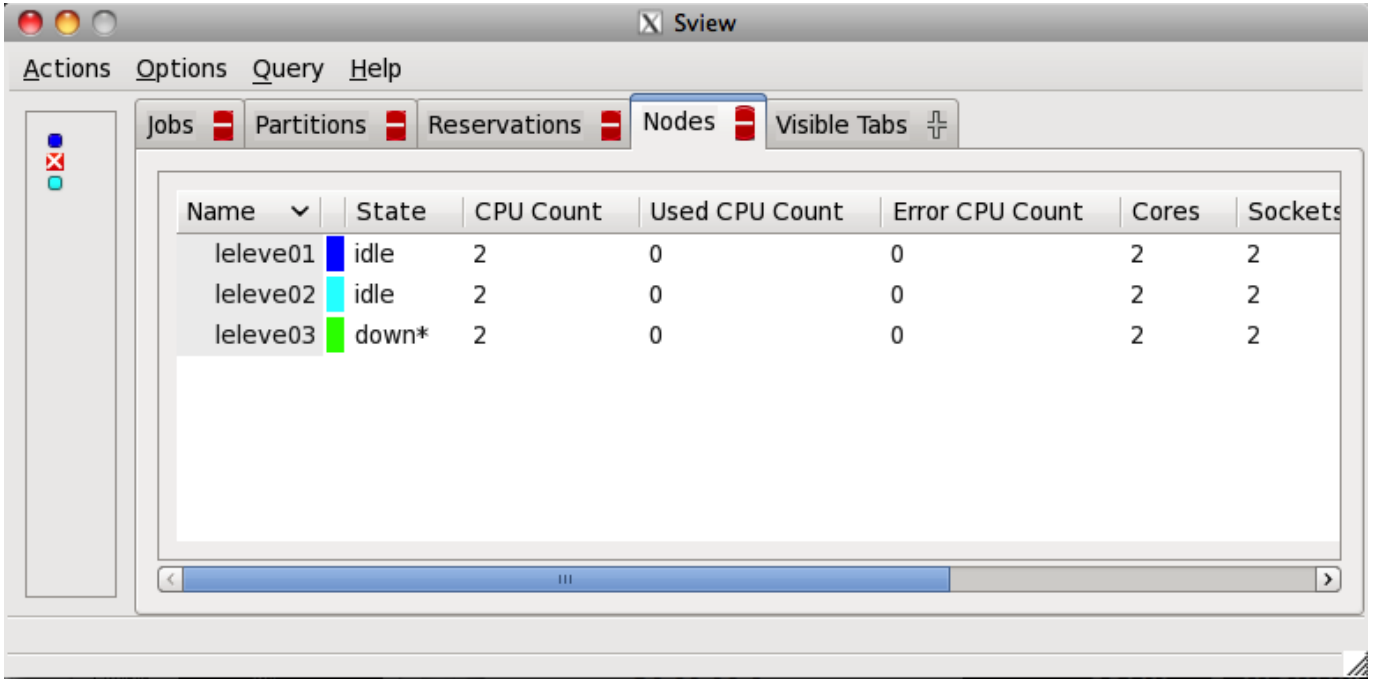
5. Control your job

- 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



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:~ #
```

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@hmem00:~/exp $ sshare -a
             Account       User Raw Shares Norm Shares   Raw Usage Effectv Usage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
root                                          1.000000   505463451      1.000000   0.500000
 root                                    1    0.000998           0      0.000000   1.000000
 others                                400    0.399202   294661668      0.582953   0.363419
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914         347      0.004256   0.363358
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914        1758      0.004259   0.363099
  others                                 1    0.002914     3321209      0.010778   0.077016
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914          20      0.004255   0.363445
  others                                 1    0.002914    20849576      0.045202   0.000021
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914     2987273      0.010122   0.090022
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
  others                                 1    0.002914           0      0.004255   0.363445
```

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

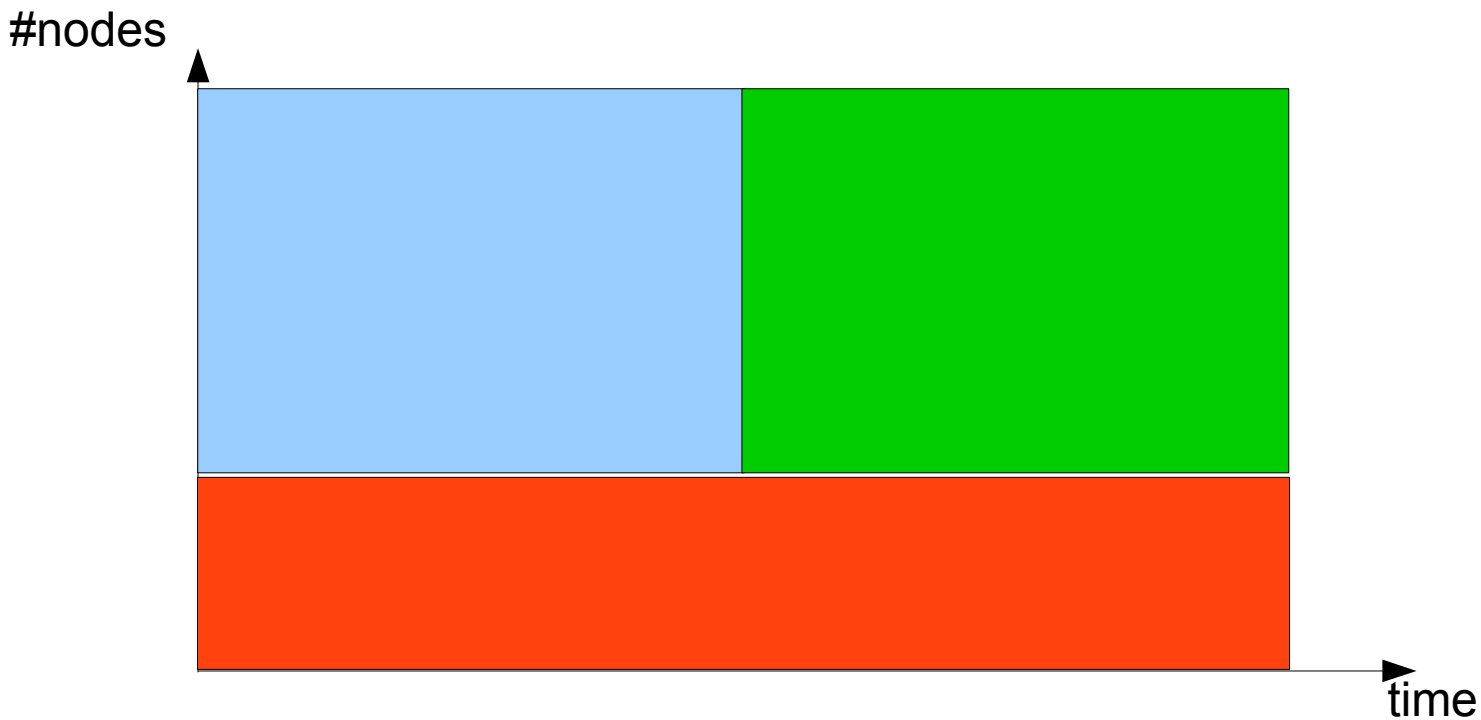
S
: is the normalized shares

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.

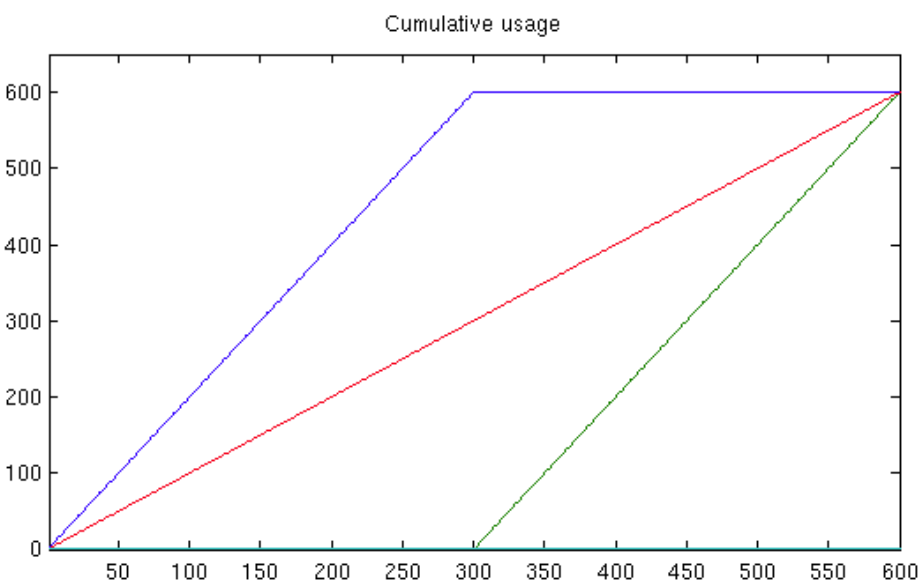
A word about fairshare

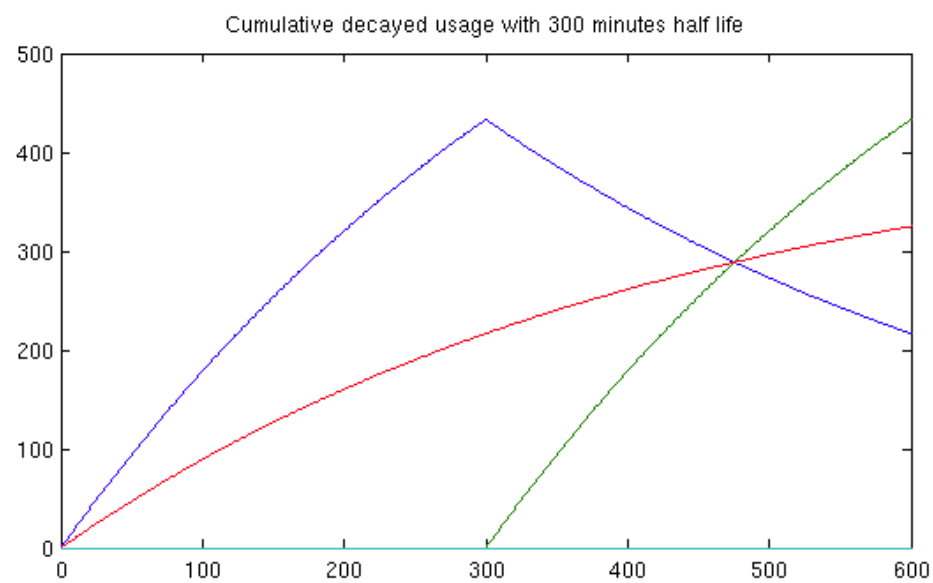
- Assume 3 users, 3-cores cluster
  - Red uses 1 core for a certain period of time
  - Blue uses 2 cores for half that period
  - Red uses 2 cores afterwards



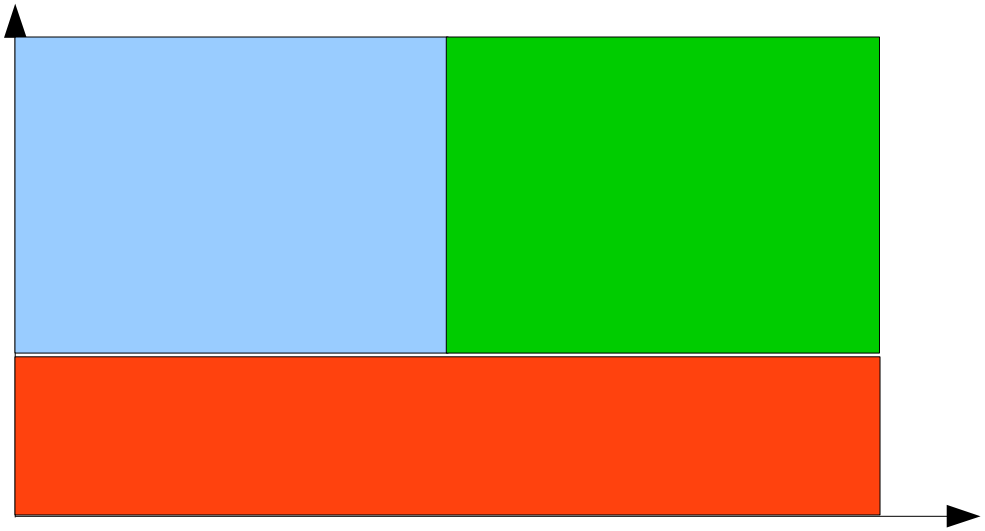
A word about fairshare

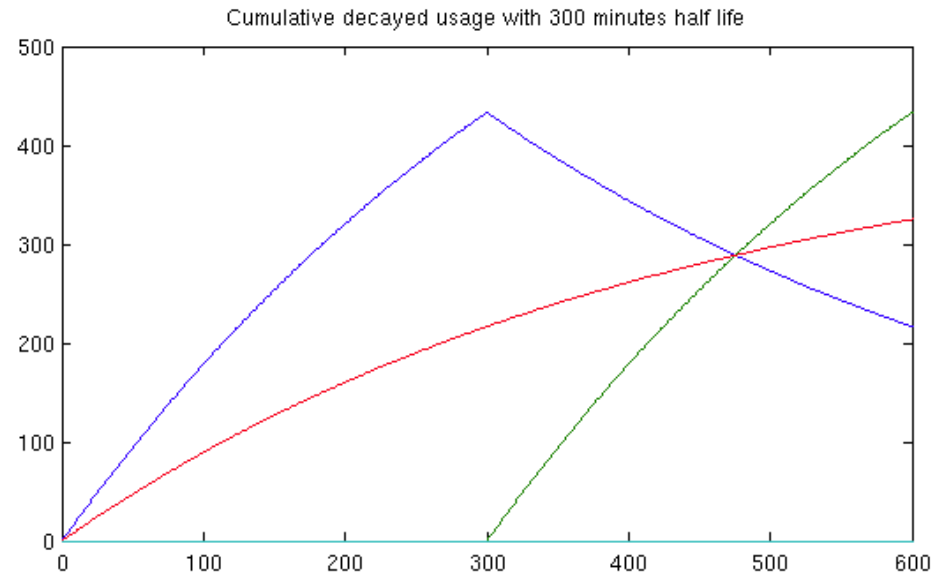
- Assume 3 users, 3-cores cluster
  - Red uses 1 core for a certain period of time
  - Blue uses 2 cores for half that period
  - Red uses 2 cores afterwards

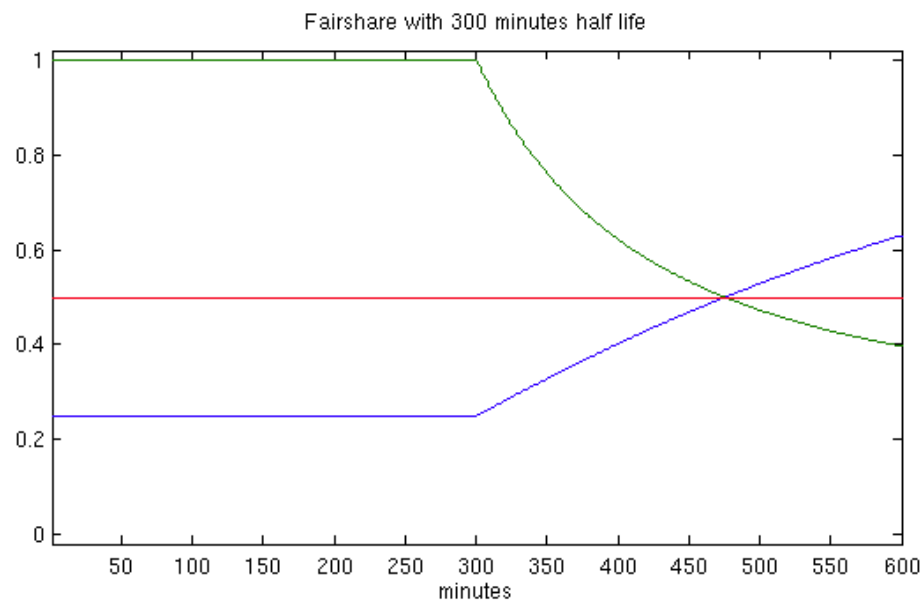




A word about fairshare







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
- sjstat

```
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:~ $
```

Getting cluster info

- sinfo
- sjstat

```
dfr@hmem00:~ $ sjstat -c

Scheduling pool data:
-------------------------------------------------------------
--------
Pool        Memory  Cpus  Total Usable   Free  Other
Traits
-------------------------------------------------------------
--------
High      512000Mb    48      2      2      0
Middle    256000Mb    48      7      7      0
Low*      128000Mb    48      8      8      0
Low*      256000Mb    48      7      7      0
Fast      128000Mb     8      3      3      0

dfr@hmem00:~ $
```

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 enviroment
  - Get node features (sinfo --node --long)
  - Get node usage (sinfo --summarize)
- Submit a job:
  - Define the resources 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?

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:

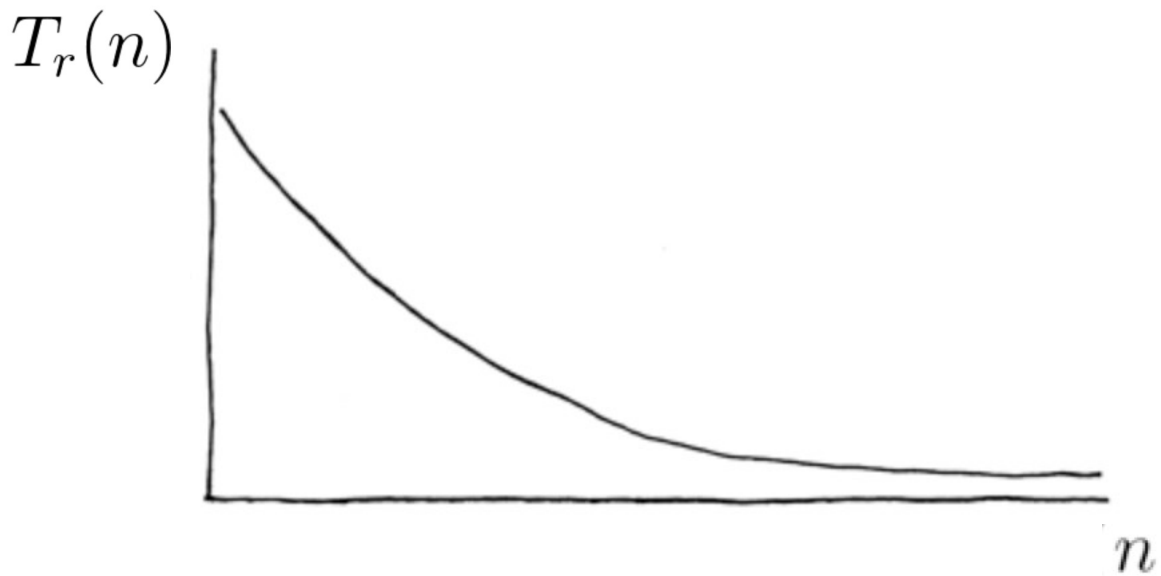
$$\mathsf{P}(T_r(n) > t) < \epsilon$$

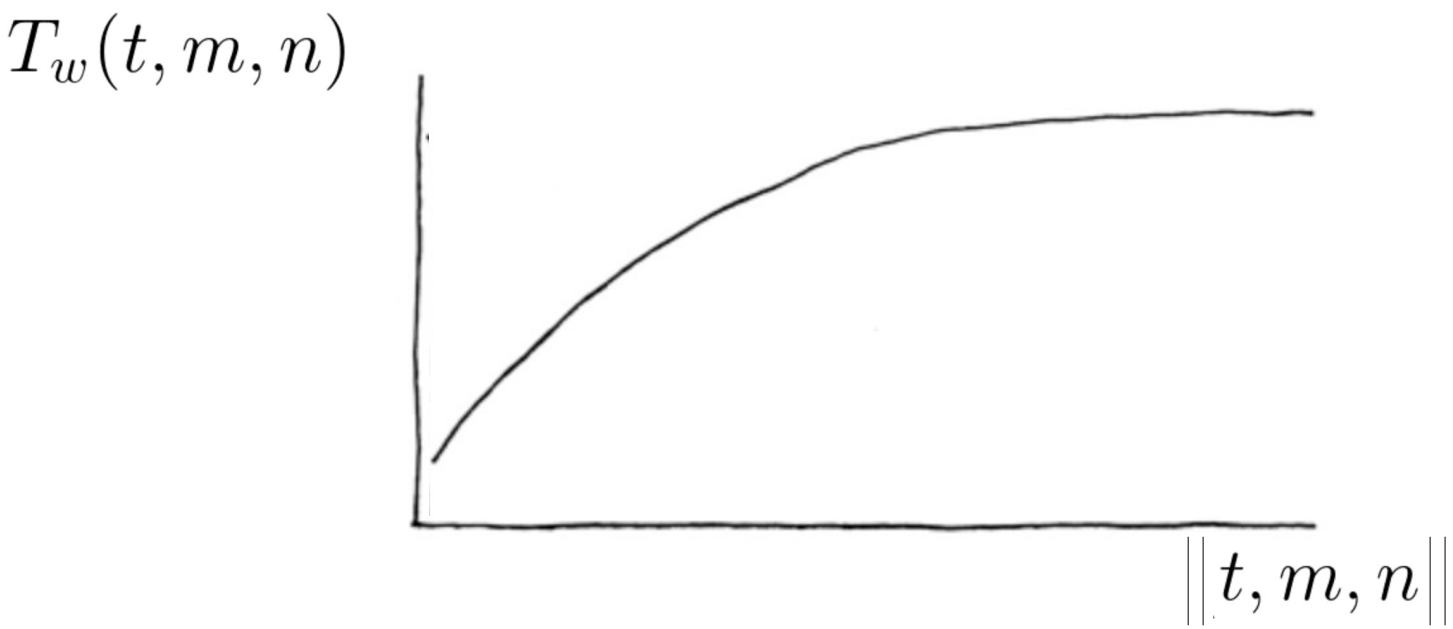
$$\mathsf{P}(M_r(n) > m) < \epsilon$$

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".

Theoretical approach

- given
  - the size of the input data,
  - the time- and space-complexity of the algorithm and
  - its strong- and weak-scaling characterization,
  - the implementation-specific additional requirements
  - the compiler-induced overhead
- you should be able to predict the resources your job needs

Practical approach

- Run a sized-down problem on your laptop or on the frontend and observe memory usage and time needed for several values of the number of CPUs.
- 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

```
[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
1357875.bat+   9383852K   1992Mc 1-23:14:58          8      200515M
1363074                   1992Mc                     8
1363074.bat+  10265148K   1992Mc 2-02:28:25          8      223659M
1363075                   1992Mc                     8
```

You will learn how to:

Create a parallel job
Request distributed resources

with

You will learn how to:

Create a parallel job
Request distributed resources

4 typical use cases:

1. MPI programs
2. Multithreaded programs
3. Master/slave
4. Embarrassingly parallel

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/1.6.4-GCC-4.7.2

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=N*
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 or --ntasks=*N* --nodes=N/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	--cpu-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 --cpu-per-task=8

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

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**, 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, **distinct files**, etc.

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

submit.sh

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

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

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

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
```

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

CECI

www.ceci-hpc.be

CÉCI Clusters News Training FAQ HowTo's Contact Create Account

C.E.C.I

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

CECI

http://www.ceci-hpc.be/scriptgen.html

Italien ■ CMU WEB SERVICE CÉCI Login CÉCI Redmine Data Scienc...ck Exchange Computation...k Exchange

All | Simple Linux Utility for... | Simple Linux Utility fo... | Simple Linux Utility fo... | Component Icons – Do... | CECI

CÉCI Clusters News Training FAQ HowTo's Contact Create Account

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
```