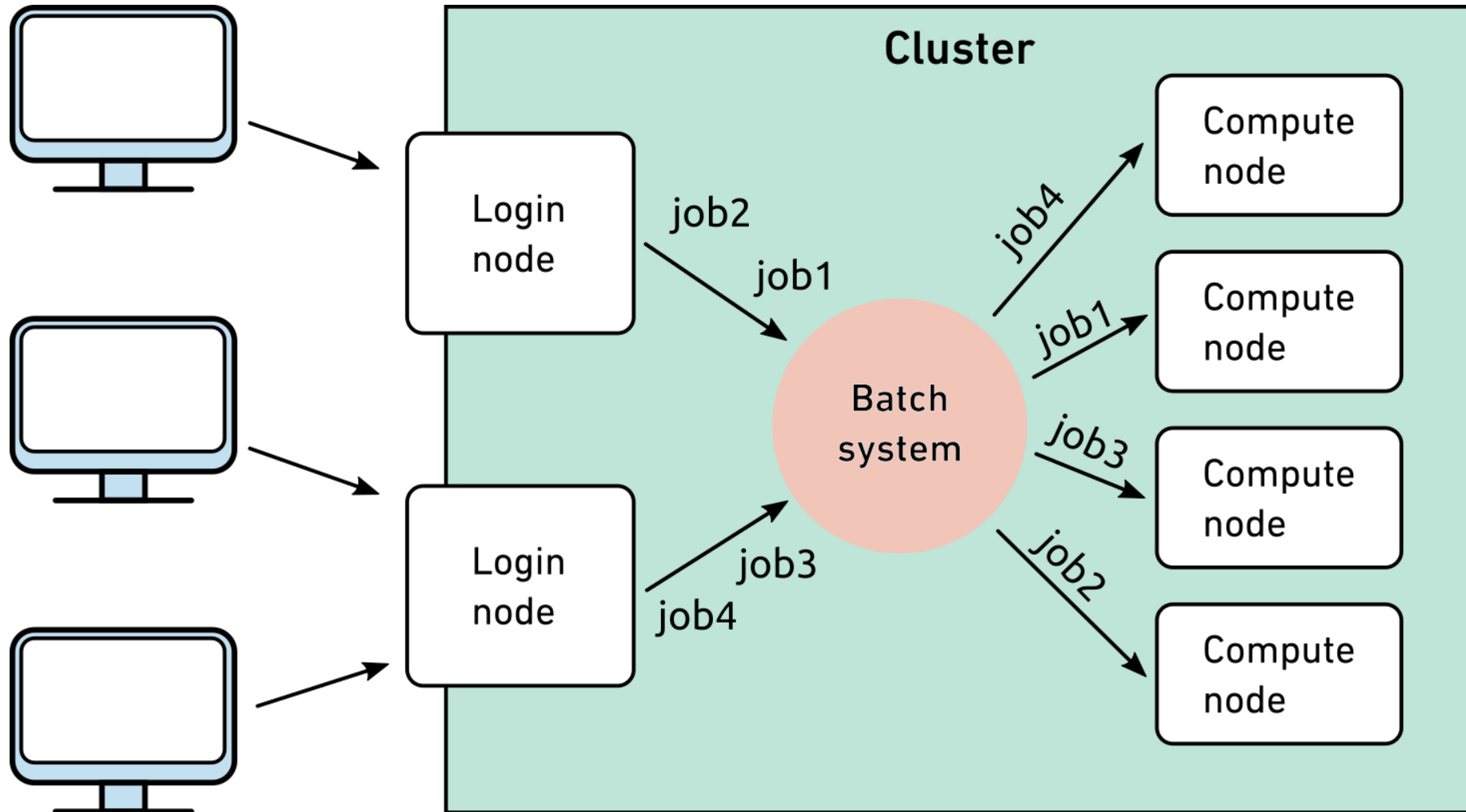


Using the batch system

Samuel Fux, Nadia Marounina
High Performance Computing Group
Scientific IT Services, ETH Zurich



Batch > Overview



Batch > Overview

- The batch system of Euler is called **Slurm** (Simple Linux Utility for Resource Management)
- Slurm manages all resources available on the cluster and allocates them to user jobs
 - Ensures that resources are used as efficiently as possible
 - Calculates user/job priorities based on a fair share principle
- All computations must be submitted to the batch system
 - There is no other way to access the cluster's compute nodes
- Please do not run computations on the login nodes
 - Login nodes may only be used for file transfer, compilation, code testing and debugging, and quick pre- and post-processing

Batch > Basic job submission

- Use `sbatch` to submit a job to the batch system
- `sbatch [Slurm options] --wrap="job"`
- A *job* can be either ...
 - a single Linux command
 - a shell script, passed via “<”
 - a [here document](#), passed via “<<”
 - a program, with its path
 - a command or program, with its arguments
 - multiple commands, enclosed in quotes
 - piped commands, enclosed in quotes
 - a command with I/O redirection, quoted
- We'll talk about `sbatch`'s options later

```
cmd
< script
<< EOF ... EOF
/path/to/program
cmd arg1 arg2
"cmd1 ; cmd2"
"cmd1 | cmd2"
"cmd <in >out"
```

Batch > Basic job submission

- When you submit a job via `sbatch`, the batch system analyzes it and dispatches it to a batch queue
 - Slurm always selects the best queue for your job
 - You can not select a queue yourself
- If all goes well, `sbatch` tells you
 - The job's unique identifier ("job ID") – e.g. "1010171"
- The jobid is important to check, monitor or terminate a job
- If you report a problem with a job (pending, running or done) to cluster support, then **always** provide the corresponding jobid

Batch > Basic job submission > Examples

```
[sfux@eu-login-03 ~]$ sbatch --wrap="echo hello"  
Submitted batch job 1010112
```

```
[sfux@eu-login-03 ~]$ sbatch < hello.sh  
Submitted batch job 1010113
```

```
[sfux@eu-login-03 ~]$ sbatch --wrap="./bin/hello"  
Submitted batch job 1010114
```

```
[sfux@eu-login-03 ~]$ sbatch --wrap="date; pwd; ls -l"  
Submitted batch job 1010115
```

```
[sfux@eu-login-03 ~]$ sbatch --wrap="du -sk /scratch > du.out"  
Submitted batch job 1010116
```

Batch > Resource requirements

- The batch system of Euler works like a black box
 - You do not need to know anything about queues, hosts, user groups, priorities, etc. to use it
 - You only need to specify the resources needed by your job
- The two most important resources are
 - Maximal run-time and the number of processors for parallel jobs
- These resources are passed to `sbatch` using options

```
sbatch --time=HH:MM:SS --ntasks=number_of_processors --warp="command"
```
- By default, a job will get 1 processor for 4 hour
 - If you need more time and/or processors, you must request them
 - Standard run-time limits are 4h, 24h, 120h

Batch > Advanced resource requirements

- Memory

- By default Slurm gives you 1000 MB of memory per processor (core)
- If you need more, you must request it
- For example, to request 2000 MB per processor (core):

```
sbatch --mem-per-cpu=2000 --wrap="command"
```

- Scratch space

- LSF does not allocate any local scratch space to batch jobs
- If your job writes temporary files into the local `/scratch` file system, you **must** request it
- For example, to request 10,000 MB of scratch space:

```
sbatch --tmp=10000 --wrap="command"
```

- If you don't specify any unit, then the integer value will be interpreted as MB. If you specify values in GB, then you need to add the suffix "g" (in the example above, you would write 2g instead of 2000)

Batch > sbatch options

<code>--ntasks=N</code>	request N cores (<code>--nodes=1</code> allocates all cores on a single node)
<code>--time=HH:MM:SS</code>	request a runtime of $HH:MM:SS$
<code>--output="filename"</code>	redirect job's standard output to <i>filename</i>
<code>--error="filename"</code>	redirect job's error messages to <i>filename</i>
<code>--mem-per-cpu=YYY</code>	request YYY MB memory per core (or add suffix "g" to specify GBs)
<code>--tmp=YYY</code>	request YYY MB of local scratch space (or add suffix "g" to specify GBs)
<code>--job-name="jobname"</code>	assign a <i>jobname</i> to the job
<code>--account="share"</code>	run job under a particular Euler share " <i>share</i> "
<code>--mail-type=BEGIN</code>	send an email when the job begins
<code>--mail-type=END,FAIL</code>	send an email when the job ends (finishes successfully or fails)

Batch > sbatch GPU options

`--gpus=N` request *N* gpus
`--gpus=MODEL:N` request *N* gpus of model *MODEL* (for instance `--gpus=rtx_3090:1`)
`--gres=gpumem:XXg` request a GPU with at least *XX* GB GPU memory

- Currently we have the following GPU models available in Slurm:
 - Nvidia GTX 1080 (`gtx_1080`)
 - Nvidia RTX 2080 Ti (`rtx_2080_ti`)
 - Nvidia RTX 3090 (`rtx_3090`)
 - Nvidia Titan RTX (`titan_rtx`)
 - Nvidia Quadro RTX6000 (`rtx_6000`)
 - Nvidia V100 (`v100-sxm2`)
 - Nvidia A100, 40 GB (`a100-pcie-40gb`)
 - Nvidia A100, 80 GB (`a100_80gb_pcie`)

Batch > Parallel job submission

Shared memory job (OpenMP)

- Runs on a single compute node
- Can use up to 24 processors
- To compile an openMP code

```
$ module load gcc/6.3.0
$ gcc -o hello_omp hello_omp.c
```

- To run an openMP code, define number of processors in `$OMP_NUM_THREADS`

```
$ module load gcc/6.3.0
$ export OMP_NUM_THREADS=8
$ sbatch --ntasks=8 ./hello_omp
```

Distributed memory job (MPI)

- Runs on multiple compute nodes
- Can use tens or even hundreds of processors
- To compile an MPI code

```
$ module load gcc/6.3.0
$ module load openmpi/4.0.2
$ mpicc -o hello_mpi hello_mpi.c
```

- Program must be launched using `mpirun`

```
$ module load gcc/6.3.0
$ module load openmpi/4.0.2
$ sbatch --ntasks=240
--wrap="mpirun hello_mpi"
```

Batch > Parallel job submission > Examples

```
[sfux@eu-login-03 ~]$ export OMP_NUM_THREADS=8

[sfux@eu-login-03 ~]$ sbatch --ntasks=8 --wrap="./hello_omp"

Submitted batch job 1010900

[sfux@eu-login-03 ~]$ unset OMP_NUM_THREADS

[sfux@eu-login-03 ~]$ module load gcc/8.2.0 openmpi/4.1.4

[sfux@eu-login-03 ~]$ sbatch -n 240 --wrap="mpirun ./hello_mpi"

Submitted batch job 1010901
```

Batch > Job array

- Multiple similar jobs can be submitted at once using a so-called “job array”
 - All jobs in an array share the same jobid
 - Use job index to distinguish between individual jobs in an array
 - Slurm stores array parameters in environment variables that can be used inside the jobs:

<code>\$_SLURM_ARRAY_TASK_COUNT</code>	Number of Slurm jobs in the array
<code>\$_SLURM_ARRAY_TASK_ID</code>	Array index of the elements in the array
<code>\$_SLURM_ARRAY_TASK_MIN</code>	Minimum index in the job array
<code>\$_SLURM_ARRAY_TASK_MAX</code>	Maximum index in the job array

- Examples:

```
sbatch --array=1-8 --wrap="echo Hello I am job \$_SLURM_ARRAY_TASK_ID out  
of \$_SLURM_ARRAY_TASK_COUNT"
```

Batch > Job array

- You can specify the range by using the format *start-end:step*
- This way you can map parameters (in a parameter study) in your job to the job array index
- It is also possible to use the jobid (%A) and the element number (%a) as part of the output or error files for the individual jobs in a job array:

```
sbatch --array=1-4 --output="out.%A.%a" --error "err.%A.%a" --wrap="..."
```

- For monitoring job arrays you can use the jobid and the element number
 - Using just the jobid will refer to all jobs in the array
 - Using `jobid_element` will refer to the single job in the array

Batch > Job array > Example

```
[sfux@eu-login-41 ~]$ sbatch --wrap="echo \"Hello, I am job \${SLURM_ARRAY_TASK_ID} of
\${SLURM_ARRAY_TASK_COUNT}\"
Submitted batch job 1424960
[sfux@eu-login-41 ~]$ squeue -j 1424960
      JOBID PARTITION      NAME      USER  ST      TIME      NODES NODELIST(REASON)
 1424960_4  normal.4h    wrap      sfux   R      0:45         1 eu-a2p-517
 1424960_3  normal.4h    wrap      sfux   R      0:45         1 eu-a2p-517
 1424960_2  normal.4h    wrap      sfux   R      0:45         1 eu-a2p-517
 1424960_1  normal.4h    wrap      sfux   R      0:45         1 eu-a2p-517
[sfux@eu-login-41 ~]$ squeue -j 1424960_4
      JOBID PARTITION      NAME      USER  ST      TIME      NODES NODELIST(REASON)
 1424960_4  normal.4h    wrap      sfux   R      0:48         1 eu-a2p-517
```

Batch > Exercise 3: Job array

Tasks	Commands
Go to the <code>hpc-examples</code> folder	<code>cd hpc-examples/job_arrays/ex1</code>
Submit a job array	<code>sbatch --array=1-4 --wrap="echo \\${SLURM_ARRAY_TASK_ID}"</code>
	<code>sbatch --array=1-4 --wrap="echo \\${(SLURM_ARRAY_TASK_ID*2)}"</code>
Load module	<code>module load gcc/6.3.0 python/3.8.5</code>
Submit a job	<code>sbatch --array=1-4 --wrap="python read_input.py"</code>

Batch > Job dependencies

- If you have a workflow or a pipeline, then sometimes tasks depend on each other
- A simple way to chain two jobs is to add the command to submit job 2 at the end of job 1
 - Not recommended because it is error-prone and may lead to infinite loops.
- A more robust solution is to use **job dependencies**
 - Store the jobid of the first job in a variable and specify the job dependency condition for the second
 - Possible dependency conditions: after, afterany, afterok, afternotok

```
[sfux@eu-login-15 ~]$ myjobid=$(sbatch --parsable --job-name="job1" --wrap="sleep 360")
[sfux@eu-login-15 ~]$ sbatch --job-name="job2" -d afterany:$myjobid --wrap="sleep 120"
Submitted batch job 1525788
[sfux@eu-login-15 ~]$ squeue
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST (REASON)
      1525788 normal.4h    job2      sfux PD      0:00      1 (Dependency)
      1525785 normal.4h    job1      sfux R      0:43      1 eu-a2p-530
[sfux@eu-login-15 ~]$
```

Batch > #SBATCH pragmas

- `sbatch` options can be specified either on the command line or inside a job script using the `#SBATCH` pragma, for example

```
#!/bin/bash
#SBATCH --ntasks=24           # 24 cores
#SBATCH --time=8:00:00      # 8-hour run-time
#SBATCH --mem-per-cpu=4000  # 4000 MB per core
cd /path/to/execution/folder
module load gcc/6.3.0 openmpi/4.0.2
mpirun myprogram arg1
```

- In this case, the script can be submitted using the “<” operator

```
$ sbatch < script
```

- `bsub` options specified on the command line override those inside the script

```
$ sbatch --ntasks=48 < script
```

Batch > Job monitoring > commands

<code>squeue</code>	check the state of a job in the queue
<code>scontrol</code>	check resource usage of a job
<code>sstat</code>	check information about a running job
<code>sacct</code>	detailed information about pending, running and finished jobs
<code>scancel</code>	kill a job

Batch > Job monitoring > squeue

- After submitting a job, the job will wait in a queue to be run on a compute node and has the pending status (PD). You can check the job status with the `squeue` command

```
[sfux@eu-login-41 ~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
1433323	normal.4h	wrap	sfux	PD	0:04	1	eu-g1-026-2
1433322	normal.4h	wrap	sfux	R	0:11	1	eu-a2p-483

- You can also check only for running jobs (R) or for pending jobs (PD):

```
[sfux@eu-login-41 ~]$ squeue -t RUNNING
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
1433322	normal.4h	wrap	sfux	R	0:28	1	eu-a2p-483

```
[sfux@eu-login-41 ~]$ squeue -t PENDING
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
1433323	normal.4h	wrap	sfux	PD	0:21	1	eu-g1-026-2

```
[sfux@eu-login-41 ~]$
```

Batch > Job monitoring > scontrol

```
[sfux@eu-login-15 ~]$ squeue -u sfux
          JOBID PARTITION    NAME     USER ST       TIME  NODES NODELIST(REASON)
          1498523 normal.4h   wrap      sfux  R        0:28     1 eu-a2p-528
[sfux@eu-login-15 ~]$ scontrol show jobid -dd 1498523
JobId=1498523 JobName=wrap
  UserId=sfux(40093) GroupId=sfux-group(104222) MCS_label=N/A
  Priority=1769 Nice=0 Account=normal/es_hpc QOS=es_hpc/normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  DerivedExitCode=0:0
  RunTime=00:00:38 TimeLimit=01:00:00 TimeMin=N/A
  SubmitTime=2022-10-27T11:44:30 EligibleTime=2022-10-27T11:44:30
  AccrueTime=2022-10-27T11:44:30
  StartTime=2022-10-27T11:44:31 EndTime=2022-10-27T12:44:31 Deadline=N/A
  SuspendTime=None SecsPreSuspend=0 LastSchedEval=2022-10-27T11:44:31 Scheduler=Main
  Partition=normal.4h AllocNode:Sid=eu-login-15:26645
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=eu-a2p-528
  BatchHost=eu-a2p-528
  NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
  TRES=cpu=1,mem=1G,node=1,billing=1
  Socks/Node=* NtasksPerN:B:S:C=0:0:*:1 CoreSpec=*
  JOB_GRES=(null)
    Nodes=eu-a2p-528 CPU_IDS=127 Mem=1024 GRES=
  MinCPUsNode=1 MinMemoryCPU=1G MinTmpDiskNode=0
  Features=(null) DelayBoot=00:00:00
  OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
  Command=(null)
  WorkDir=/cluster/home/sfux
  StdErr=/cluster/home/sfux/slurm-1498523.out
  StdIn=/dev/null
  StdOut=/cluster/home/sfux/slurm-1498523.out
  Power=
```

Batch > Job monitoring > sacct

- The command `sstat` provides you information about running jobs:

```
[sfux@eu-login-15 ~]$ sacct --format JobID,State,AllocCPUS,Elapsed,NNodes,NTasks,TotalCPU,REQMEM,MaxRSS,NodeList -j 1525037
```

JobID	State	AllocCPUS	Elapsed	NNodes	NTasks	TotalCPU	ReqMem	MaxRSS	NodeList
1525037	RUNNING	4	00:01:27	1		00:00:00	16000M		eu-a2p-373
1525037.bat+	RUNNING	4	00:01:27	1	1	00:00:00			eu-a2p-373
1525037.ext+	RUNNING	4	00:01:27	1	1	00:00:00			eu-a2p-373

- Possible format options for `sacct`:

Account,AdminComment,AllocCPUS,AllocNodes,AllocTRES,AssocID,AveCPU,AveCPUFreq,AveDiskRead,AveDiskWrite,AvePages,AveRSS,AveVMSize,BlockID,Cluster,Comment,Constraints,ConsumedEnergy,ConsumedEnergyRaw,Container,CPUTime,CPUTimeRAW,DBIndex,DerivedExitCode,Elapsed,ElapsedRaw,Eligible,End,ExitCode,Flags,GID,Group,JobID,JobIDRaw,JobName,Layout,MaxDiskRead,MaxDiskReadNode,MaxDiskReadTask,MaxDiskWrite,MaxDiskWriteNode,MaxDiskWriteTask,MaxPages,MaxPagesNode,MaxPagesTask,MaxRSS,MaxRSSNode,MaxRSSTask,MaxVMSize,MaxVMSizeNode,MaxVMSizeTask,McsLabel,MinCPU,MinCPUNode,MinCPUTask,NCPUS,NNodes,NodeList,NTasks,Partition,Priority,QOS,QOSRAW,Reason,ReqCPUFreq,ReqCPUFreqGov,ReqCPUFreqMax,ReqCPUFreqMin,ReqCPUS,ReqMem,ReqNodes,ReqTRES,Reservation,ReservationId,Reserved,ResvCPU,ResvCPURAW,Start,State,Submit,SubmitLine,Suspended,SystemComment,SystemCPU,Timelimit,TimelimitRaw,TotalCPU,TRESUsageInAve,TRESUsageInMax,TRESUsageInMaxNode,TRESUsageInMaxTask,TRESUsageInMin,TRESUsageInMinNode,TRESUsageInMinTask,TRESUsageInTot,TRESUsageOutAve,TRESUsageOutMax,TRESUsageOutMaxNode,TRESUsageOutMaxTask,TRESUsageOutMin,TRESUsageOutMinNode,TRESUsageOutMinTask,TRESUsageOutTot,UID,User,UserCPU,WCKey,WCKeyID,WorkDir

Batch > Job monitoring > myjobs

```
[sfux@eu-login-35 ~]$ myjobs -j 2039738
Job information
Job ID           : 2039738
Status          : RUNNING
Running on node  : eu-a2p-415
User            : sfux
Shareholder group : es_hpc
Queue           : normal.4h
Command         : sbatch --ntasks=4 --time=1:00:00
--mem-per-cpu=2048
Working directory :
/cluster/home/sfux/testrun/adf/2021_test
Requested resources
Requested runtime      : 01:00:00
Requested cores (total) : 4
Requested nodes       : 1
Requested memory (total) : 8192 MiB
Requested scratch (per node) : #not yet implemented#
Job history
Submitted at         : 2022-11-08T14:51:37
Started at          : 2022-11-08T14:51:38
Queue waiting time  : 1 sec
Resource usage
Wall-clock          : 00:10:57
Total CPU time      : 00:43:25
CPU utilization     : #not yet implemented#
Sys/Kernel time    : #not yet implemented#
Total resident memory : 1218.30 MiB
Resident memory utilization : 14.87%
[sfux@eu-login-35 ~]$
```

- We are working on providing a bbjobs like wrapper for monitoring Slurm jobs. The wrapper script is called myjobs and accepts a single option -j to specify the jobid
- Memory utilization has been added recently
- CPU utilization is still work in progress
- Sys/Kernel time is still work in progress
- Showing the amount of requested scratch space is still work in progress

Batch > Job monitoring > scancel

```
[sfux@eu-login-15 ~]$ squeue
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST (REASON)
      1525589 normal.24  sbatch    sfux  R      0:11      1 eu-a2p-373
[sfux@eu-login-15 ~]$ scancel 1525589
[sfux@eu-login-15 ~]$ squeue
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST (REASON)
[sfux@eu-login-15 ~]$
```

Options:

job-ID

kill *job-ID*

--name=*jobname*

kill all jobs with name *jobname*

--user=*username*

kill all jobs from user *username*

--state=*state*

kill all jobs in state *state* (states: PENDING, RUNNING or SUSPENDED)

Batch > Job output

- Each Slurm job creates a file `Slurm-JOBID.out` in the directory where you submitted the job from
- The Slurm log file contains the stdout and stderr of your job if you did not redirect it
- If a job fails, then you can find the error message in the `Slurm-JOBID.out` file
- If you report a problem about a job to cluster support, then always provide the corresponding Slurm file with the job output

Dos and don'ts

Dos

- Understand what you are doing
- Ask for help if you don't understand what you are doing
- Optimize your workflow to make it as efficient as possible
- Keep in mind that our clusters are shared by many users
- Choose the file system you want to use carefully

Don'ts

- Don't waste CPU time or disk space
- Don't run applications on the login nodes
- Don't write large amounts of data to standard output
- Don't create millions of small files
- Don't run hundreds of small jobs if the same work can be done in a single job

Getting help

- Wiki: <https://scicomp.ethz.ch>
- Ticket system
 - <https://smartdesk.ethz.ch> (ETH account authentication)
 - Please describe your problem as accurately as possible
- E-mail
 - cluster-support@id.ethz.ch
 - Please do not send questions to individual members of the team
- Person-to-person (not during COVID 19)
 - Contact us to set up an appointment at your place
 - Visit us at Binzmühlestrasse 130

Questions?