Submitting and monitoring jobs on the HPC

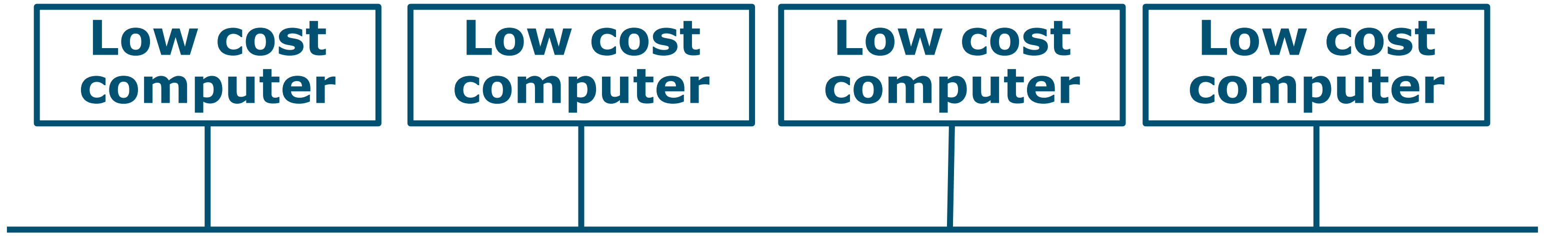
J. Vandenplas, H.J. Megens, G. Dawes

June 7, 2017

WAGENINGEN UR
For quality of life

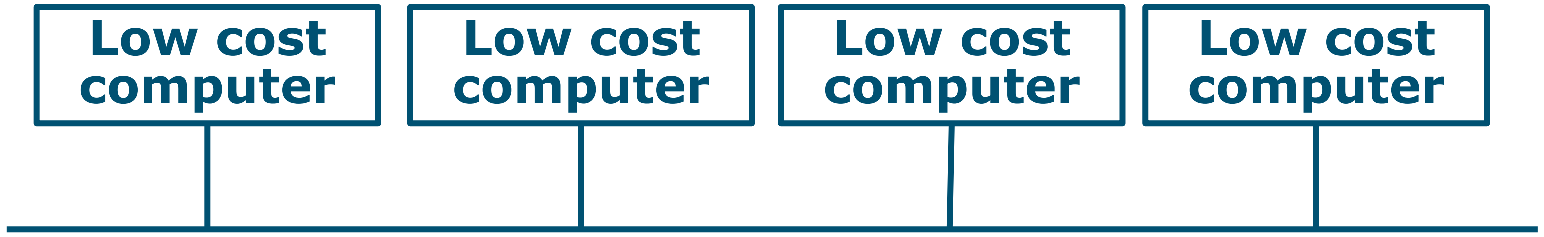
Some definitions

- High performance computing cluster
    - Group of interconnected computers (node) that work together and act like a single system



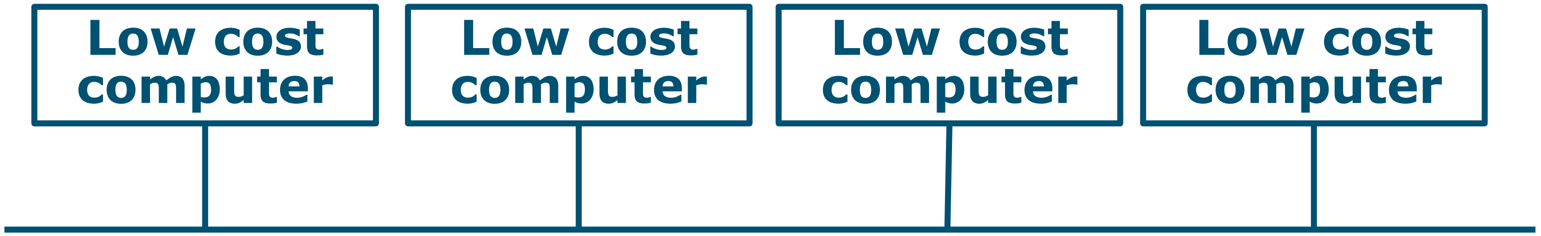
Some definitions

- High performance computing cluster
  - Group of interconnected computers (node) that work together and act like a single system
- CPU (Central processing unit)
  - Component within a computer that carries out the instructions of a computer program



Some definitions

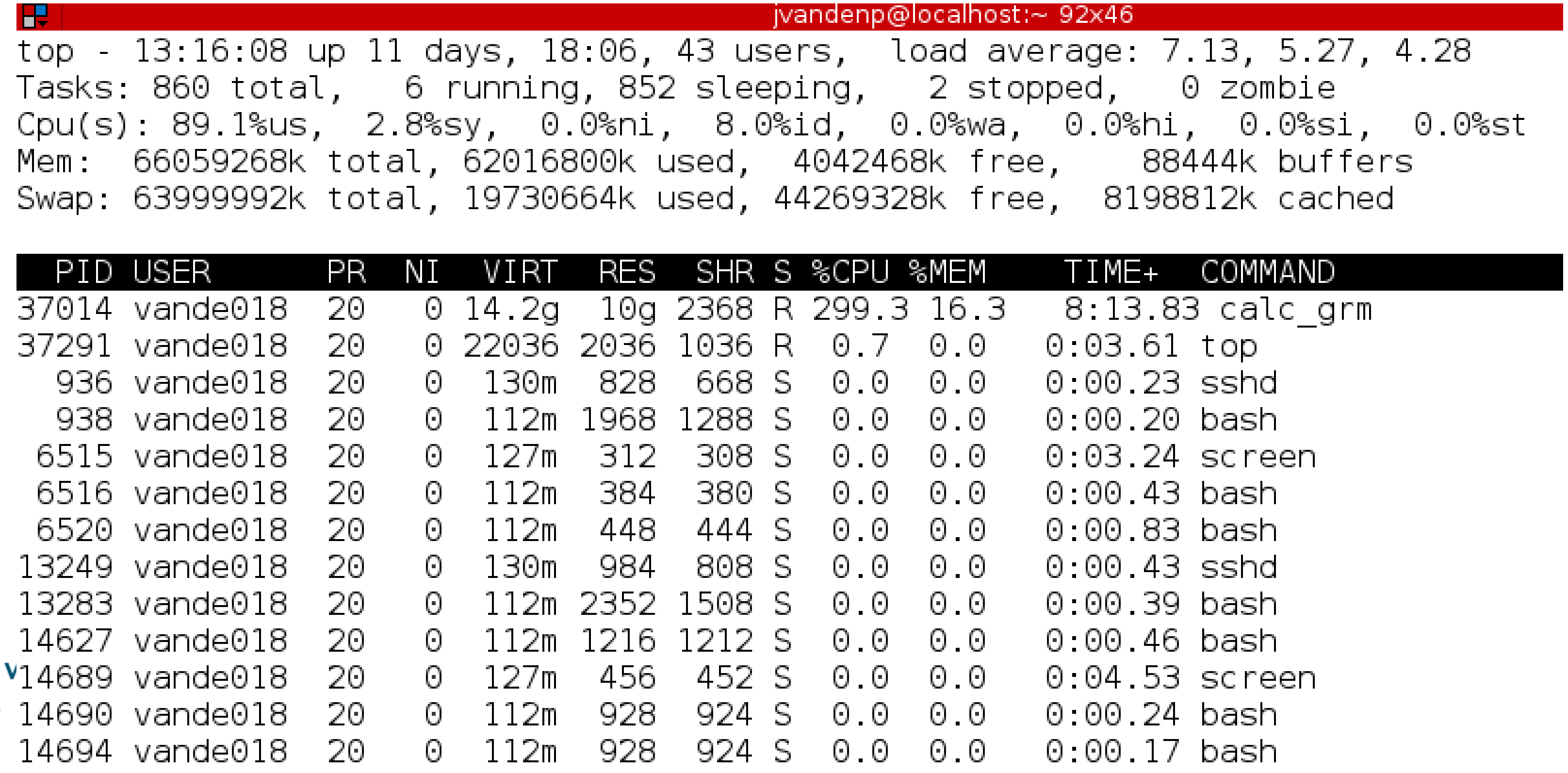
- High performance computing cluster
  - Group of interconnected computers (node) that work together and act like a single system
- CPU (Central processing unit)
  - Component within a computer that carries out the instructions of a computer program
- Core
  - Processing unit which reads and executes program instructions



WAGENINGEN UR
For quality of life

Some definitions

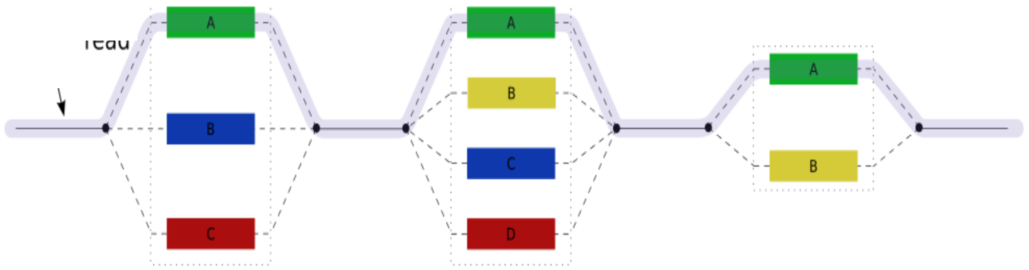
- Process
  - Instance of a computer program that is being executed



```
jvandenp@localhost:~ 92x46
top - 13:16:08 up 11 days, 18:06, 43 users,  load average: 7.13, 5.27, 4.28
Tasks: 860 total,   6 running, 852 sleeping,   2 stopped,   0 zombie
Cpu(s): 89.1%us,  2.8%sy,  0.0%ni,  8.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  66059268k total, 62016800k used,  4042468k free,    88444k buffers
Swap: 63999992k total, 19730664k used, 44269328k free,  8198812k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
37014 vande018  20   0 14.2g  10g 2368 R 299.3 16.3   8:13.83 calc_grm
37291 vande018  20   0 22036 2036 1036 R  0.7  0.0   0:03.61 top
  936 vande018  20   0  130m  828  668 S  0.0  0.0   0:00.23 sshd
  938 vande018  20   0  112m 1968 1288 S  0.0  0.0   0:00.20 bash
 6515 vande018  20   0  127m  312  308 S  0.0  0.0   0:03.24 screen
 6516 vande018  20   0  112m  384  380 S  0.0  0.0   0:00.43 bash
 6520 vande018  20   0  112m  448  444 S  0.0  0.0   0:00.83 bash
13249 vande018  20   0  130m  984  808 S  0.0  0.0   0:00.43 sshd
13283 vande018  20   0  112m 2352 1508 S  0.0  0.0   0:00.39 bash
14627 vande018  20   0  112m 1216 1212 S  0.0  0.0   0:00.46 bash
14689 vande018  20   0  127m  456  452 S  0.0  0.0   0:04.53 screen
14690 vande018  20   0  112m  928  924 S  0.0  0.0   0:00.24 bash
14694 vande018  20   0  112m  928  924 S  0.0  0.0   0:00.17 bash
```

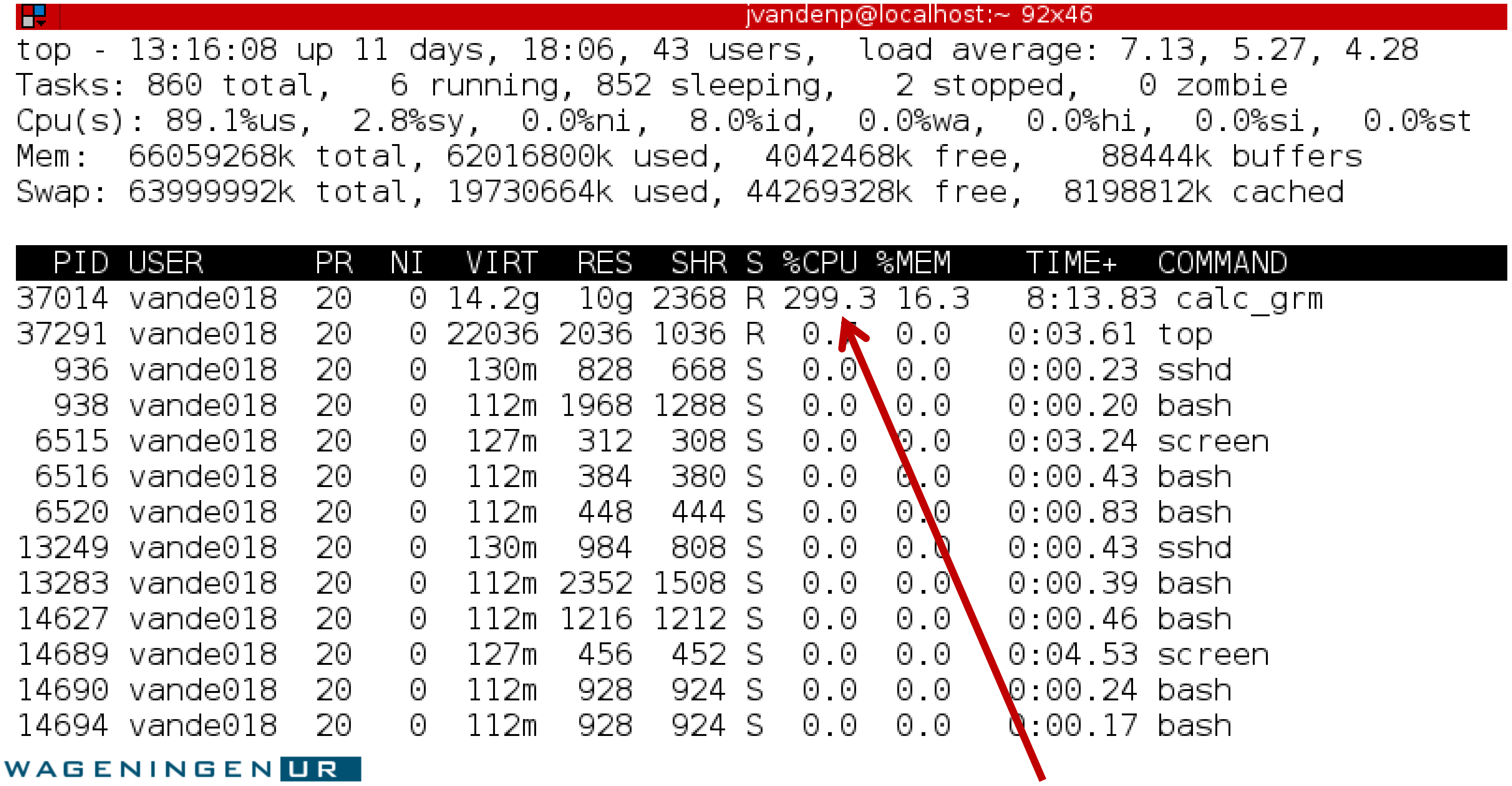
Some definitions

- Process
  - Instance of a computer program that is being executed
  - May be made up of multiple threads that execute instructions concurrently
- Thread
  - Smallest sequence of programmed instructions



Some definitions

- Process / Thread
  - Linux command: ***top***

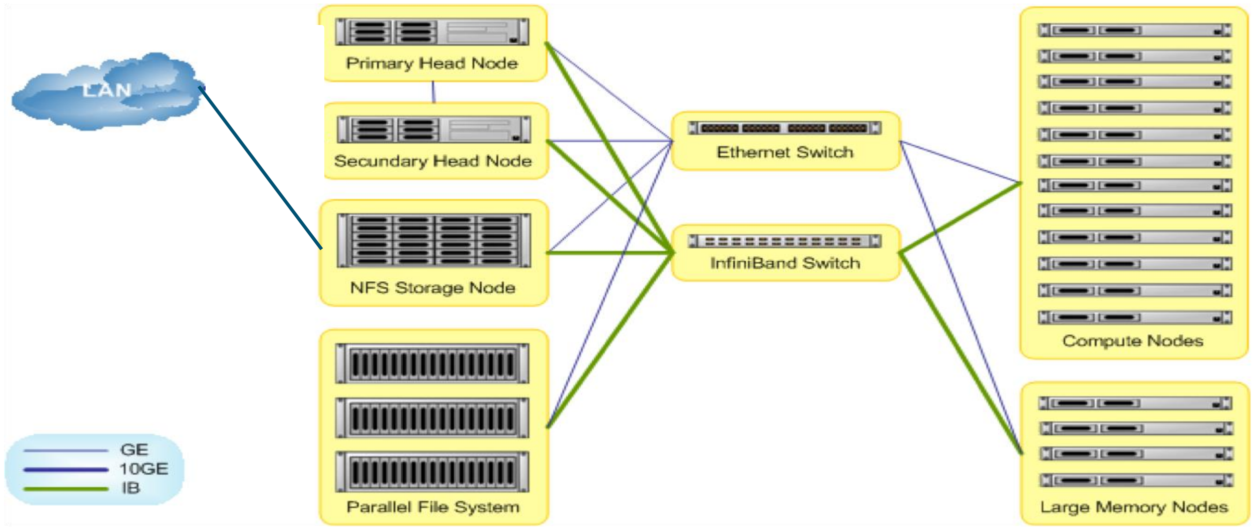


jvandenp@localhost:~ 92x46

```
top - 13:16:08 up 11 days, 18:06, 43 users,  load average: 7.13, 5.27, 4.28
Tasks: 860 total,   6 running, 852 sleeping,   2 stopped,   0 zombie
Cpu(s): 89.1%us,  2.8%sy,  0.0%ni,  8.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  66059268k total, 62016800k used,  4042468k free,    88444k buffers
Swap: 63999992k total, 19730664k used, 44269328k free,  8198812k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
37014 vande018  20   0 14.2g  10g 2368 R 299.3 16.3   8:13.83 calc_grm
37291 vande018  20   0 22036 2036 1036 R  0.7  0.0   0:03.61 top
  936 vande018  20   0  130m  828  668 S  0.0  0.0   0:00.23 sshd
  938 vande018  20   0  112m 1968 1288 S  0.0  0.0   0:00.20 bash
 6515 vande018  20   0  127m  312  308 S  0.0  0.0   0:03.24 screen
 6516 vande018  20   0  112m  384  380 S  0.0  0.0   0:00.43 bash
 6520 vande018  20   0  112m  448  444 S  0.0  0.0   0:00.83 bash
13249 vande018  20   0  130m  984  808 S  0.0  0.0   0:00.43 sshd
13283 vande018  20   0  112m 2352 1508 S  0.0  0.0   0:00.39 bash
14627 vande018  20   0  112m 1216 1212 S  0.0  0.0   0:00.46 bash
14689 vande018  20   0  127m  456  452 S  0.0  0.0   0:04.53 screen
14690 vande018  20   0  112m  928  924 S  0.0  0.0   0:00.24 bash
14694 vande018  20   0  112m  928  924 S  0.0  0.0   0:00.17 bash
```

Agrogenomics HPC

- 2 head nodes
- Compute nodes
  - 48 nodes (16 cores; 64GB RAM)
  - 2 fat nodes (64 cores; 1TB RAM)



Running a job on the nodes of the HPC?

WAGENINGEN UR
For quality of life

Running a job on the nodes of the HPC?

- Job
  - An operation or a group of operations treated as a single and distinct unit
  - Two parts
    - Resource requests
    - Job steps
      - Tasks that must be done (e.g., software that must be run)

Running a job on the nodes of the HPC?

- Job
    - An operation or a group of operations treated as a single and distinct unit
    - Two parts
        - Resource requests
        - Job steps
            - Tasks that must be done (e.g., software that must be run)
- A job must be submitted to a job scheduler

  ➔Requires a (shell) submission script

Job scheduler/Resource manager

- Software which:
  - Manages and allocates resources (computer nodes)
  - Manages and schedules jobs on a set of allocated nodes
  - Sets up the environment for parallel and distributed computing

Job scheduler/Resource manager

- Software which:
  - Manages and allocates resources (compute nodes)
  - Manages and schedules jobs on a set of allocated nodes
  - Sets up the environment for parallel and distributed computing
- HPC's job scheduler: **SLURM** (Simple Linux Utility for Resource Management ; http://slurm.schedmd.com/slurm.html)

Some definitions for Slurm

- Task
    - In the Slurm context, it must be understood as a process.

Some definitions for Slurm

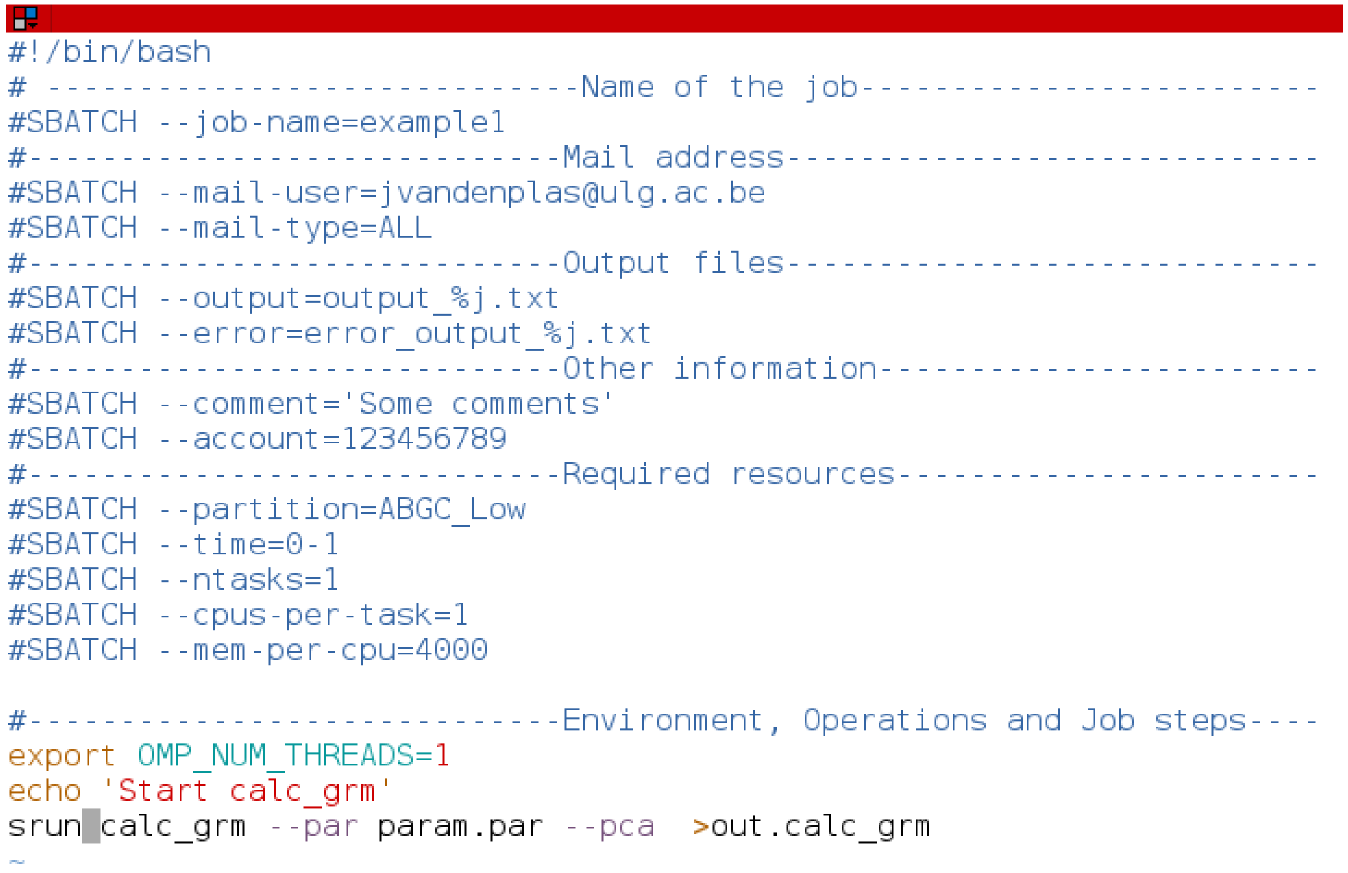
- Task
  - In the Slurm context, it must be understood as a process.
- CPU
  - In the Slurm context, it can be understood as a core or a hardware thread.

Some definitions for Slurm

- Task
    - In the Slurm context, it must be understood as a process.
- CPU
    - In the Slurm context, it can be understood as a core or a hardware thread.
- Multithreaded program
    - One task using several CPUs
- Multi-process program
    - Several tasks

Running a job on the nodes of the HPC?

- A submission script is required...



```
#!/bin/bash
# -----------------------------Name of the job------------------------
#SBATCH --job-name=example1
#-----------------------------Mail address-----------------------------
#SBATCH --mail-user=jvandenplas@ulg.ac.be
#SBATCH --mail-type=ALL
#-----------------------------Output files-----------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=123456789
#-----------------------------Required resources-----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=0-1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
export OMP_NUM_THREADS=1
echo 'Start calc_grm'
srun calc_grm --par param.par --pca  >out.calc_grm
~
```

➔... and it must be submitted!

Running a job on the nodes of the HPC?

Several steps

1. Characteristics of the jobs?
2. Writing a submission script
3. Submitting a job
4. Monitoring and controlling a job
5. Getting an overview of previous and current jobs

1. Characteristics of the job

- What is your job?
  - Sequential/parallel
  - Resource requests
    - Number of CPUs
    - Amount of RAM
    - Expected computing time
    - ...
  - Jobs steps
    - Job steps can be created with the command ***srun***

WAGENINGEN UR
For quality of life

1. Characteristics of the job

- Try to fit to the real use as much as possible!
- Try to ask 4GB RAM per CPU for the compute node (15.6GB RAM per CPU for the large memory nodes)

1. Characteristics of the job

- What is your job?
  - Sequential/parallel
  - If parallel: multi-process vs multi-threaded?

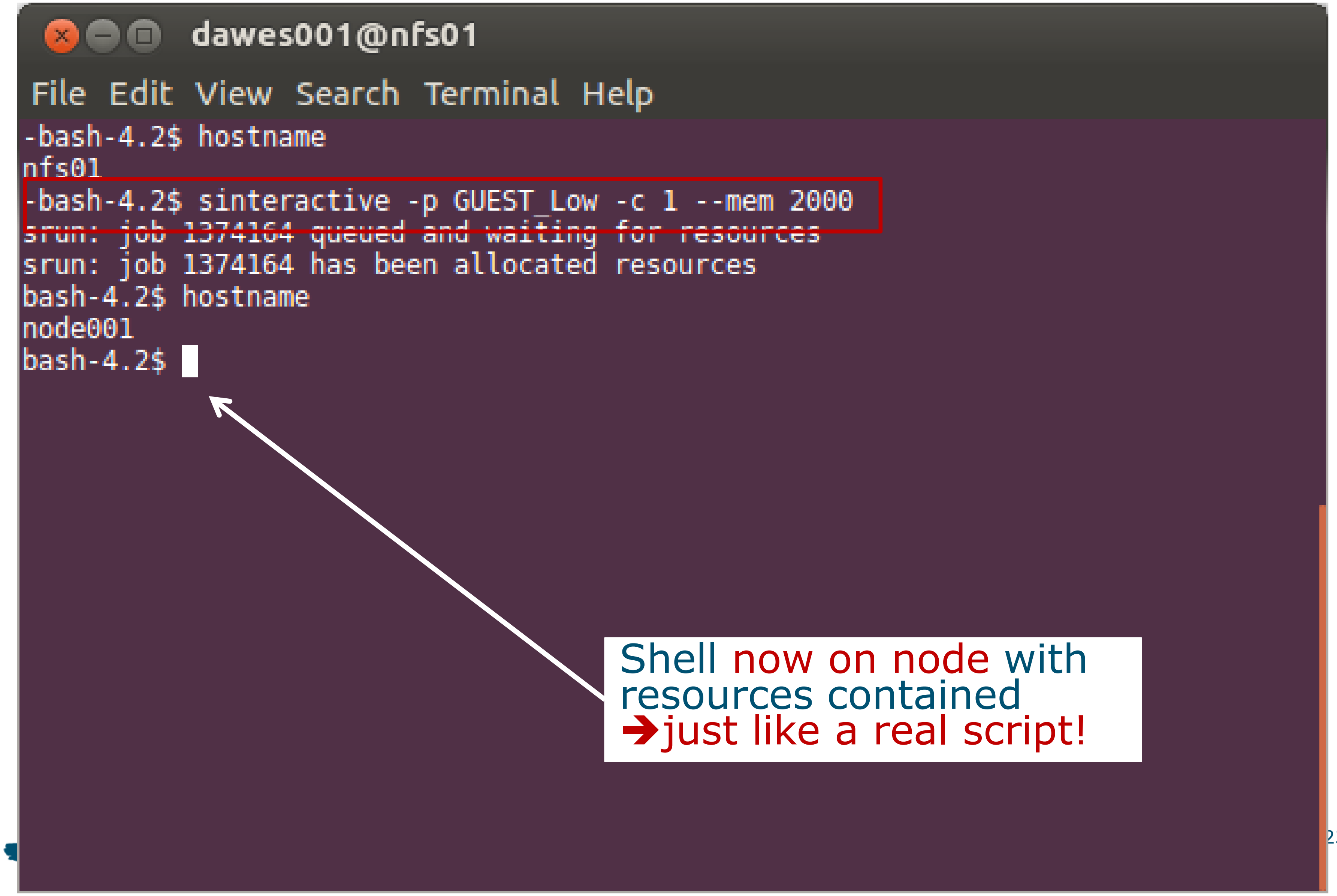
➔How can you tell?
  - RTFM!
  - Read the source code (if available)
  - Just run it!

    ➔use ***sinteractive***!

1. Characteristics of the job

- Run the job using Sandbox environment – interactive jobs
    - ***sinteractive***
        - Wrapper on ***srun***
        - Request immediate interactive shell on node(s)
    - ***sinteractive*** –p GUEST_LOW –c <cpus> --mem <MB>

1. Characteristics of the job



```
-bash-4.2$ hostname
nfs01
-bash-4.2$ sinteractive -p GUEST_Low -c 1 --mem 2000
srun: job 1374164 queued and waiting for resources
srun: job 1374164 has been allocated resources
bash-4.2$ hostname
node001
bash-4.2$ 
```

Shell now on node with resources contained
➔just like a real script!

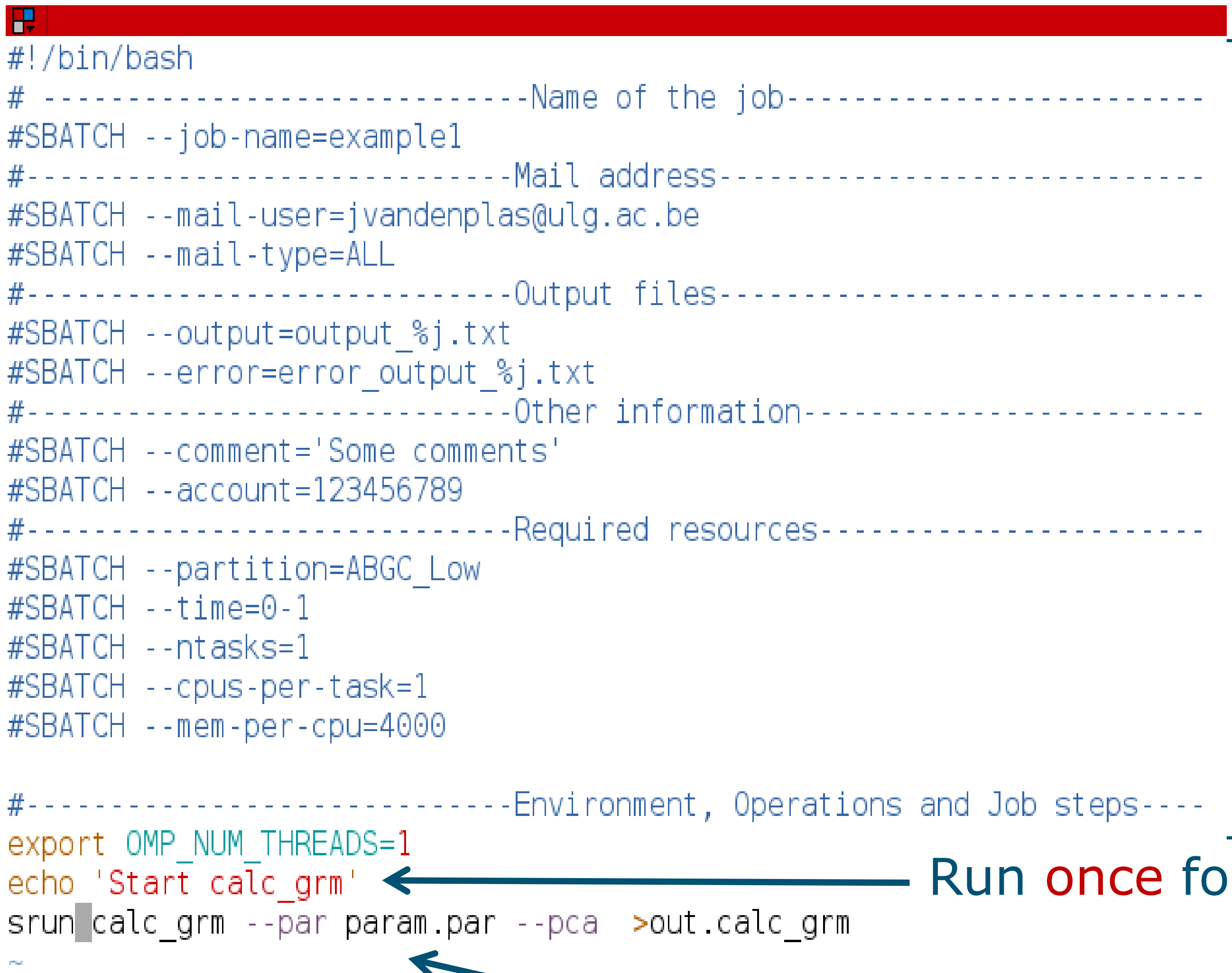
Try it...

- Copy the following directory (e.g., in your $HOME)
    - /lustre/shared/training_slurm/sinteractive

- Try to find the requirements (CPUs, memory).

WAGENINGEN UR
For quality of life

2. Writing a submission script



```
#!/bin/bash
# ------------------------------Name of the job-------------------------
#SBATCH --job-name=example1
#-----------------------------Mail address-------------------------------
#SBATCH --mail-user=jvandenplas@ulg.ac.be
#SBATCH --mail-type=ALL
#-----------------------------Output files-------------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information--------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=123456789
#-----------------------------Required resources-------------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=0-1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
export OMP_NUM_THREADS=1
echo 'Start calc_grm'
srun calc_grm --par param.par --pca  >out.calc_grm
~
```

SLURM options

Run once for a single task

Run for each task

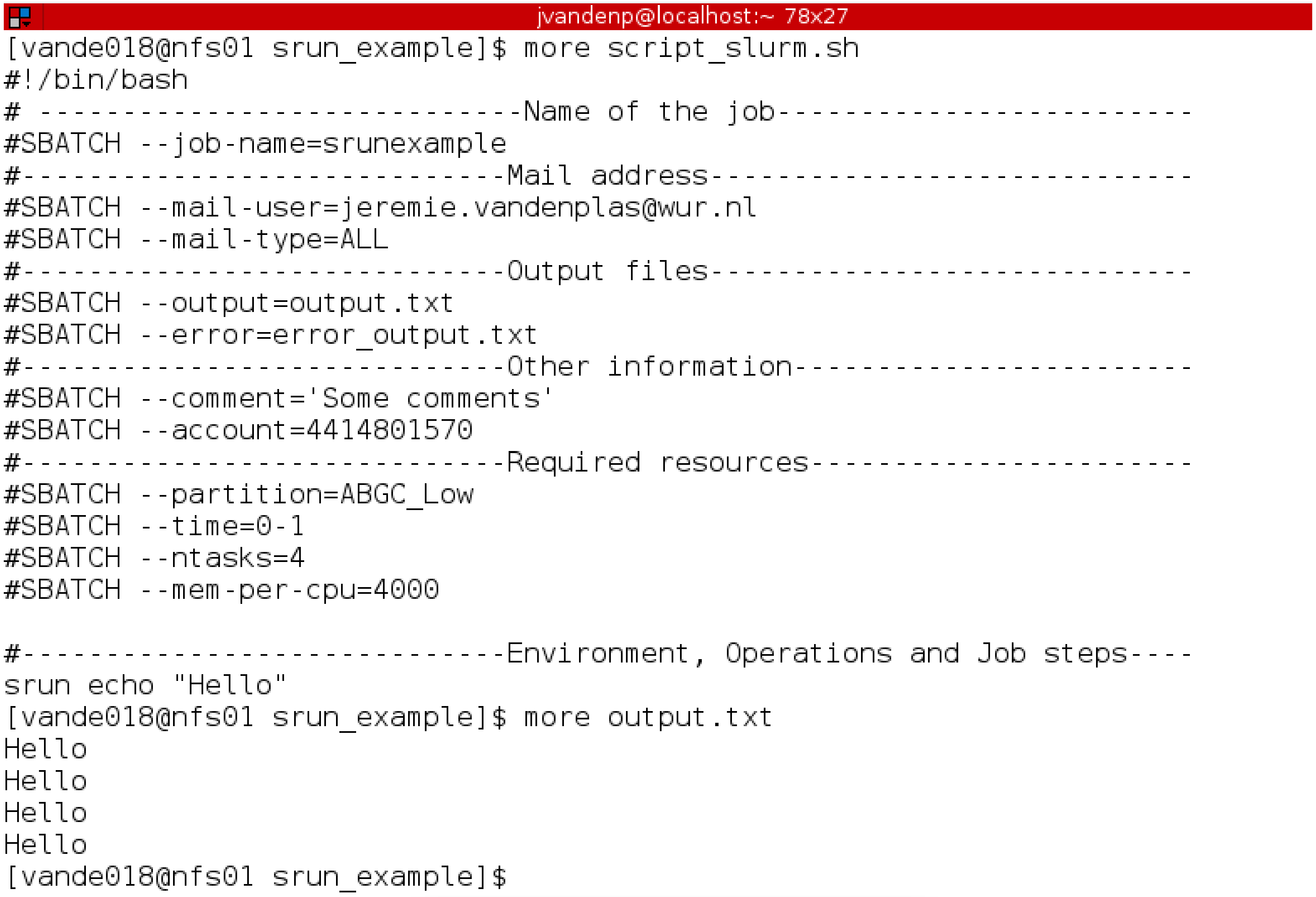
The Slurm command *srun*

- ***srun*** [options] executable [args]
    - Run a parallel job on cluster
    - Useful options

Option	Report
-c=<ncpus>	Request that ncpus allocated per process
-n=<number>	Specify the number of tasks to run

WAGENINGEN UR
For quality of life

The Slurm command *srun*



jvandenp@localhost:~ 78x27

```
[vande018@nfs01 srun_example]$ more script_slurm.sh
#!/bin/bash
# -----------------------------Name of the job-------------------------
#SBATCH --job-name=srunexample
#-----------------------------Mail address-----------------------------
#SBATCH --mail-user=jeremie.vandenplas@wur.nl
#SBATCH --mail-type=ALL
#-----------------------------Output files-----------------------------
#SBATCH --output=output.txt
#SBATCH --error=error_output.txt
#-----------------------------Other information------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=4414801570
#-----------------------------Required resources-----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=0-1
#SBATCH --ntasks=4
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
srun echo "Hello"
[vande018@nfs01 srun_example]$ more output.txt
Hello
Hello
Hello
Hello
[vande018@nfs01 srun_example]$
```

WAGENINGEN UR
For quality of life

Some SLURM options

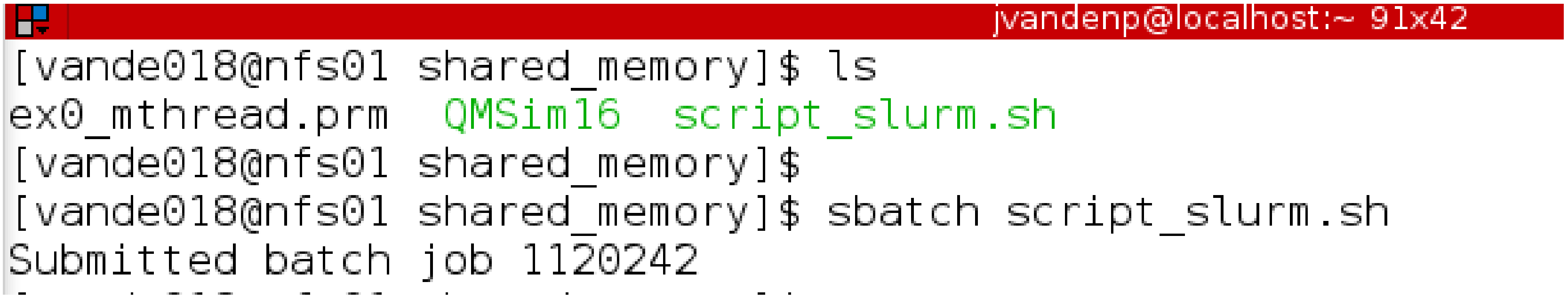
You want	**SLURM option**
To set a **job name**	--job-name=“job1”
To get **emails**	--mail-user=name.name@wur.nl --mail-type=BEGIN|END|FAILED|ALL
To set the name of the **output files**	--output=output_%j.txt --error=error_output_%j.txt
To set the name of an **account**	--account=12345678
To attach a **comment** to the job	--comment=“abcd”

Some SLURM options: resource

You want	SLURM option
To choose a **partition**	--partition=ABGC_Low|Std|High
To choose a **specific feature** (e.g., a regular compute node)	--constraint=normalmem|largemem
3 **independent processes**	--ntasks=3
3 **independent processes** to spread across **2 nodes**	--ntasks=3 --ntasks-per-node=2
3 **processes** that can use each **2 cores**	--ntasks=3 --cpus-per-task=2
4000MB per cpu	--mem-per-cpu=4000

3. Submitting a job

- The scripts are submitted using the ***sbatch*** command



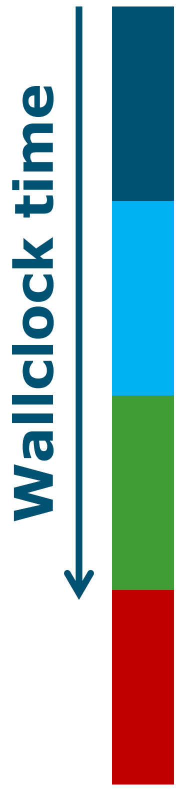
```
jvandenp@localhost:~ 91x42
[vande018@nfs01 shared_memory]$ ls
ex0_mthread.prm  QMSim16  script_slurm.sh
[vande018@nfs01 shared_memory]$
[vande018@nfs01 shared_memory]$ sbatch script_slurm.sh
Submitted batch job 1120242
```

- Slurm gives an ID to the job ($JOBID)
- Options may be passed from the command line
  - E.g., sbatch --ntasks=3 script_slurm.sh
  - Will override value in script

Some jobs and their option requirements

- Serial example
- Embarrassingly parallel example
- Shared memory example
- Message passing example

A serial example



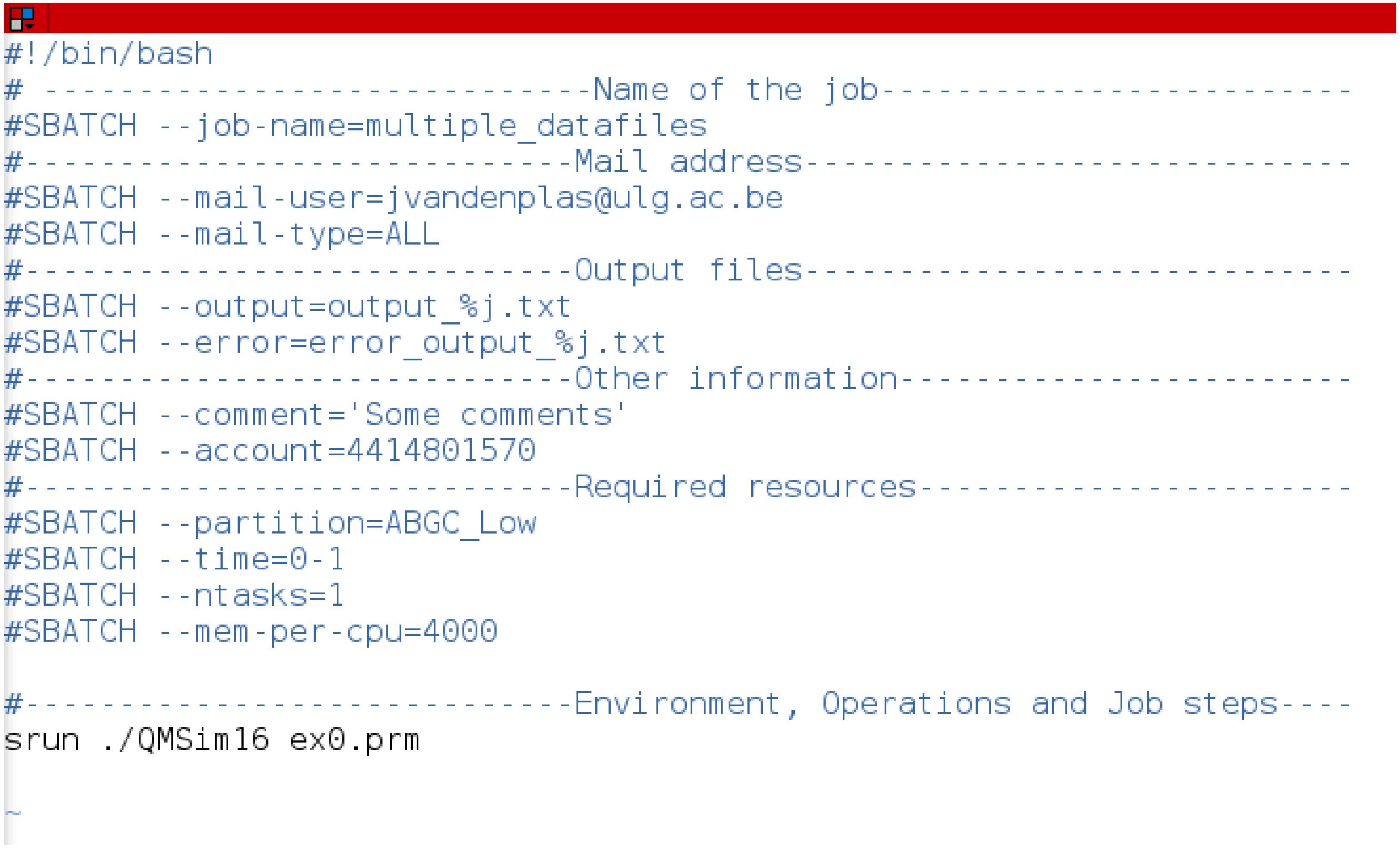
- You run one (several) program(s) serially
- There is no parallelism

A serial example: resource

You want	SLURM options
To chose a partition	--partition=ABGC_Std
8 hours	--time=00-08:00:00
1 independent process	--ntasks=1
4000MB per CPU	--mem-per-cpu=4000
You use	(srun) ./myprog

WAGENINGEN UR
For quality of life

A serial example: script



```
#!/bin/bash
# ----------------------------Name of the job-------------------------
#SBATCH --job-name=multiple_datafiles
#-----------------------------Mail address----------------------------
#SBATCH --mail-user=jvandenplas@ulg.ac.be
#SBATCH --mail-type=ALL
#-----------------------------Output files----------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information-----------------------
#SBATCH --comment='Some comments'
#SBATCH --account=4414801570
#-----------------------------Required resources----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=0-1
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
srun ./QMSim16 ex0.prm

~
```

WAGENINGEN UR
For quality of life

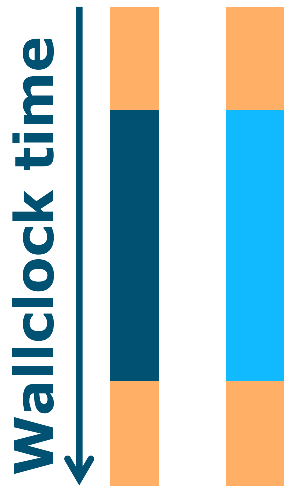
Try it...

- Copy the following directory (e.g., in your $HOME)
    - /lustre/shared/training_slurm/serial/training

- Write a Slurm script to run **onve** the script **helloworld.sh** with **1 thread**.

- Generic Slurm script
    - /lustre/shared/training_slurm/script_slurm.sh

An embarrassingly parallel example



- Parallelism is obtained by launching the same program multiple times simultaneously
- Everybody does the same thing
- No inter-process communication
- Useful cases
  - Multiple input/data files
  - Random sampling
  - ...

An embarrassingly parallel example
Multiple input/data files

- The program processes input/data from one file

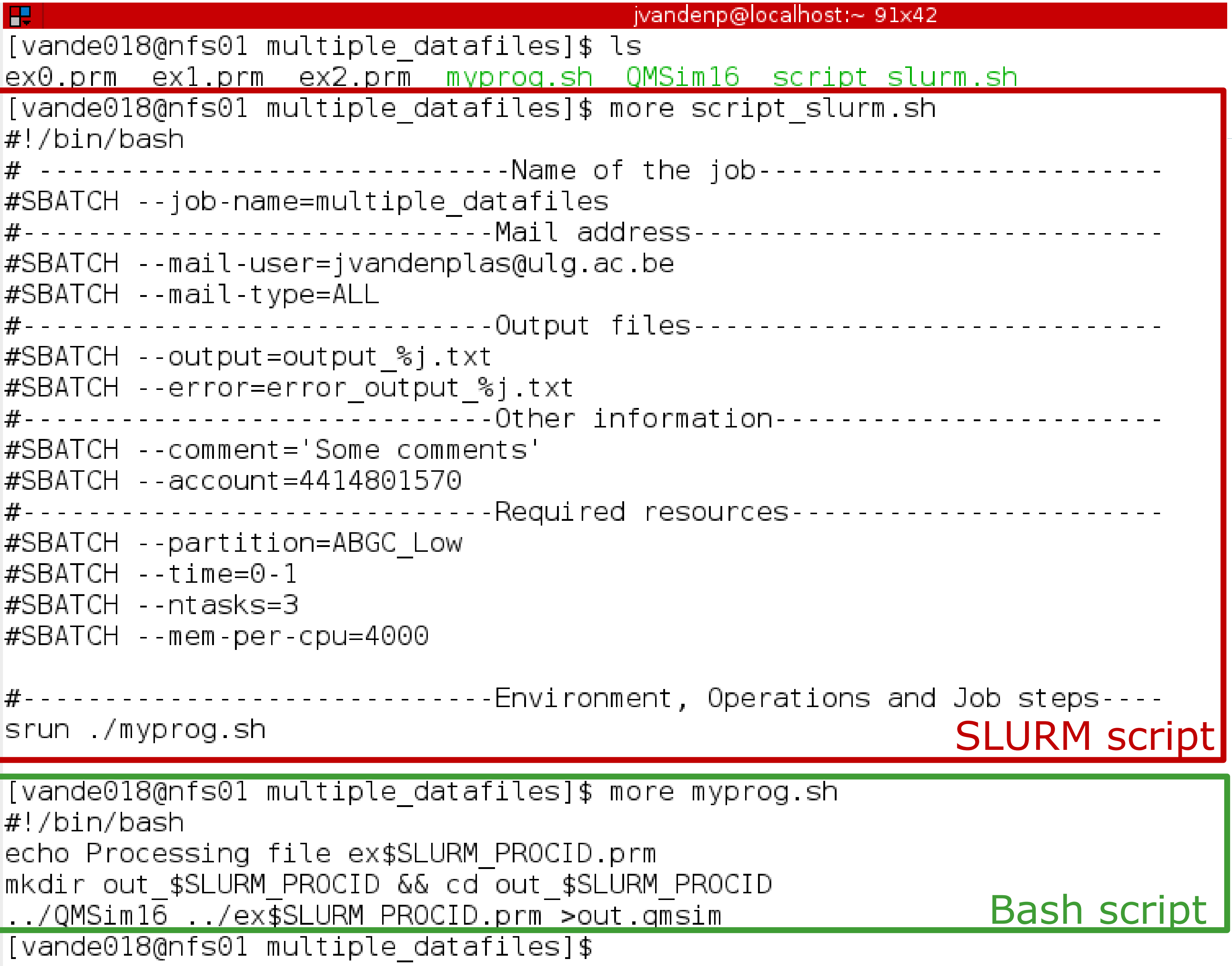
  ➔Launch the same program multiple times on distinct input/data files

- Tip: SLURM_PROCID
  - Environment variable
  - Relative process ID of the current process
  - Starts from 0 until n-1

An embarrassingly parallel example
Multiple input/data files: resource

You want	SLURM options
To chose a partition	--partition=ABGC_Std
8 hours	--time=00-08:00:00
3 independent process	--ntasks=3
4000MB per CPU	--mem-per-cpu=4000
You use	srun ./myprog $SLURM_PROCID

WAGENINGEN UR
For quality of life



```
jvandenp@localhost:~ 91x42
[vande018@nfs01 multiple_datafiles]$ ls
ex0.prm  ex1.prm  ex2.prm  myprog.sh  QMSim16  script_slurm.sh
[vande018@nfs01 multiple_datafiles]$ more script_slurm.sh
#!/bin/bash
# ----------------------------Name of the job----------------------------
#SBATCH --job-name=multiple_datafiles
#-----------------------------Mail address-----------------------------
#SBATCH --mail-user=jvandenplas@ulg.ac.be
#SBATCH --mail-type=ALL
#-----------------------------Output files-----------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information----------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=4414801570
#-----------------------------Required resources-----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=0-1
#SBATCH --ntasks=3
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
srun ./myprog.sh
```

SLURM script

```
[vande018@nfs01 multiple_datafiles]$ more myprog.sh
#!/bin/bash
echo Processing file ex$SLURM_PROCID.prm
mkdir out_$SLURM_PROCID && cd out_$SLURM_PROCID
../QMSim16 ../ex$SLURM_PROCID.prm >out.qmsim
```

Bash script

```
[vande018@nfs01 multiple_datafiles]$
```

An embarrassingly parallel example
Multiple input/data files

- The program processes input/data from one file

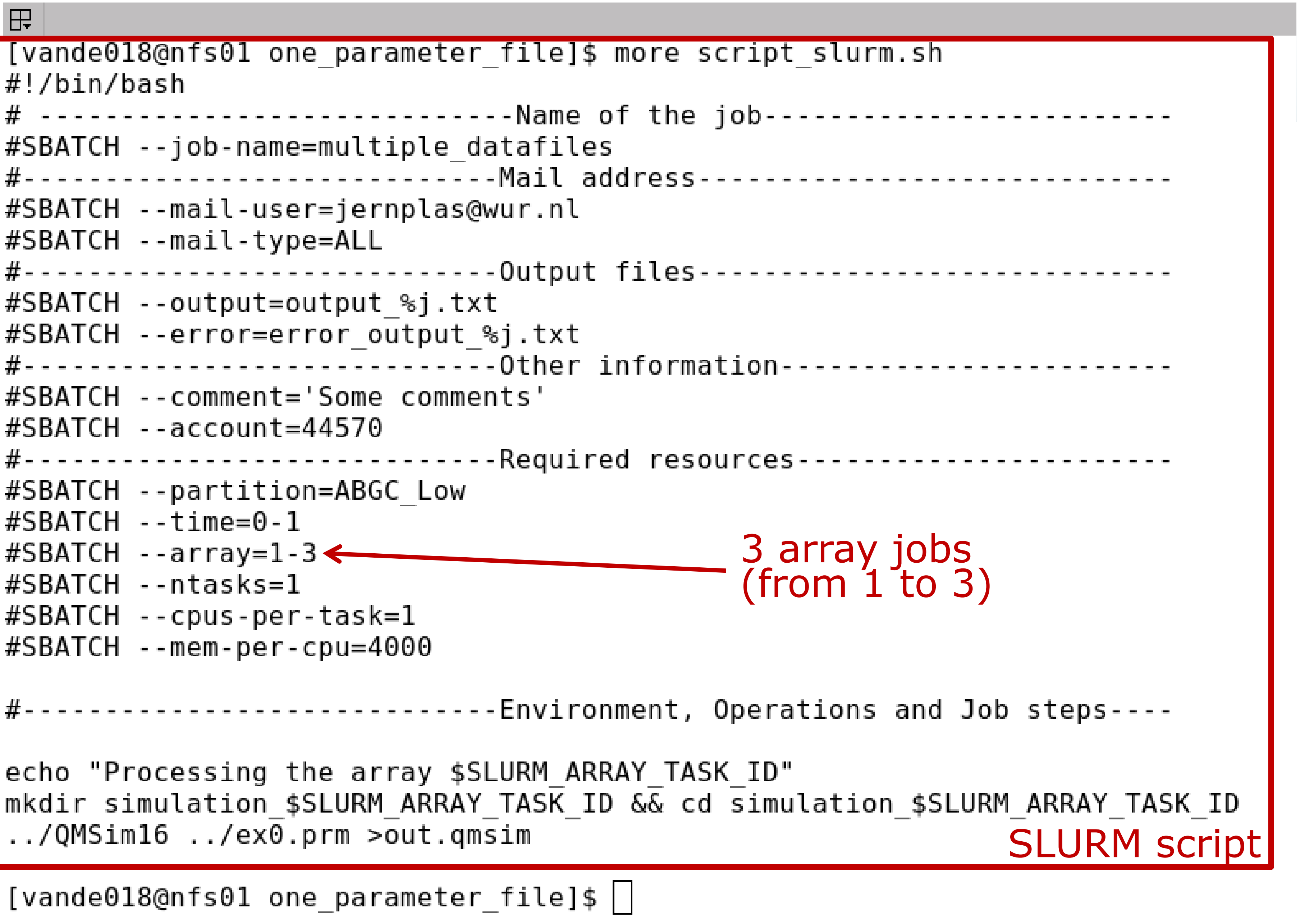
  ➔Launch the same program multiple times on distinct input/data files

- Tip: SLURM_PROCID
  - Environment variable
  - Relative process ID of the current process
  - Starts from 0 until n-1
- **Or** use job arrays!

An embarrassingly parallel example
Resource

You want	SLURM options
To chose a partition	--partition=ABGC_Std
8 hours	--time=00-08:00:00
3 processes to launch 3 completely independent jobs	--array=1-3
1 process per array	--ntasks=1
4000MB per CPU	--mem-per-cpu=4000
You use	$SLURM_ARRAY_TASK_ID (srun) ./myprog

WAGENINGEN UR
For quality of life

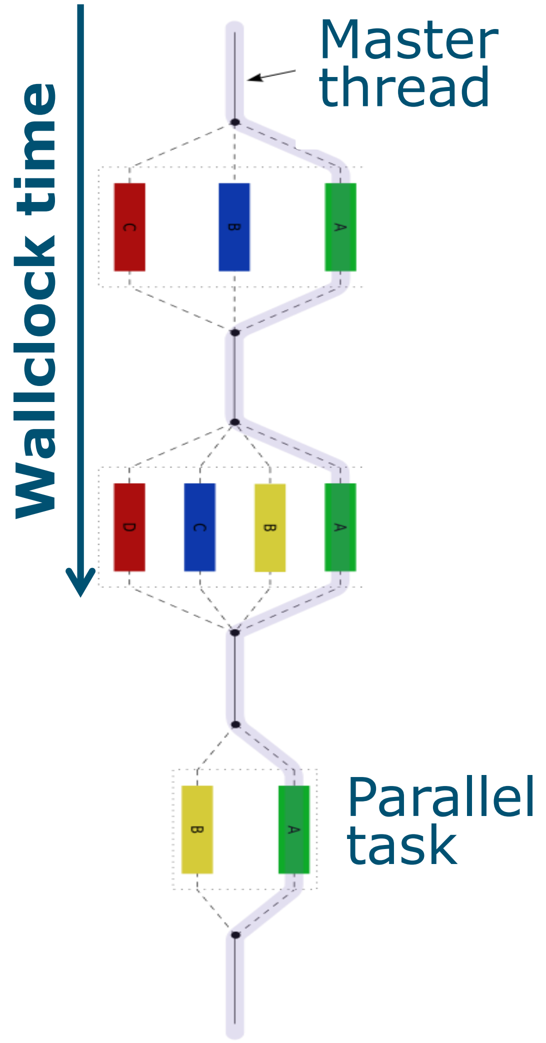


```
[vande018@nfs01 one_parameter_file]$ more script_slurm.sh
#!/bin/bash
# -----------------------------Name of the job-------------------------
#SBATCH --job-name=multiple_datafiles
#-----------------------------Mail address-----------------------------
#SBATCH --mail-user=jernplas@wur.nl
#SBATCH --mail-type=ALL
#-----------------------------Output files-----------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=44570
#-----------------------------Required resources-----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=0-1
#SBATCH --array=1-3
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----

echo "Processing the array $SLURM_ARRAY_TASK_ID"
mkdir simulation_$SLURM_ARRAY_TASK_ID && cd simulation_$SLURM_ARRAY_TASK_ID
../QMSim16 ../ex0.prm >out.qmsim
[vande018@nfs01 one_parameter_file]$
```

Try it...

- Copy the following directory in your $HOME
    - /lustre/shared/training_slurm/embarrasing_parallel/jobarray/training
- Write a Slurm script to run **4 times** the program **QMSim16** with **1 thread** and a total of **4 GB RAM**. Parameter files are numbered from ex**0**.prm to ex**3**.prm.
- Other examples
    - /lustre/shared/training_slurm/embarrasing_parallel

A shared memory example



- Parallelism is obtained by launching a multithreaded program
  - E.g., using OpenMP or TBB
- The program spawns itself on the node
- Generally run job on a single node
  - The threads cannot be split across several nodes
- Communication by shared memory

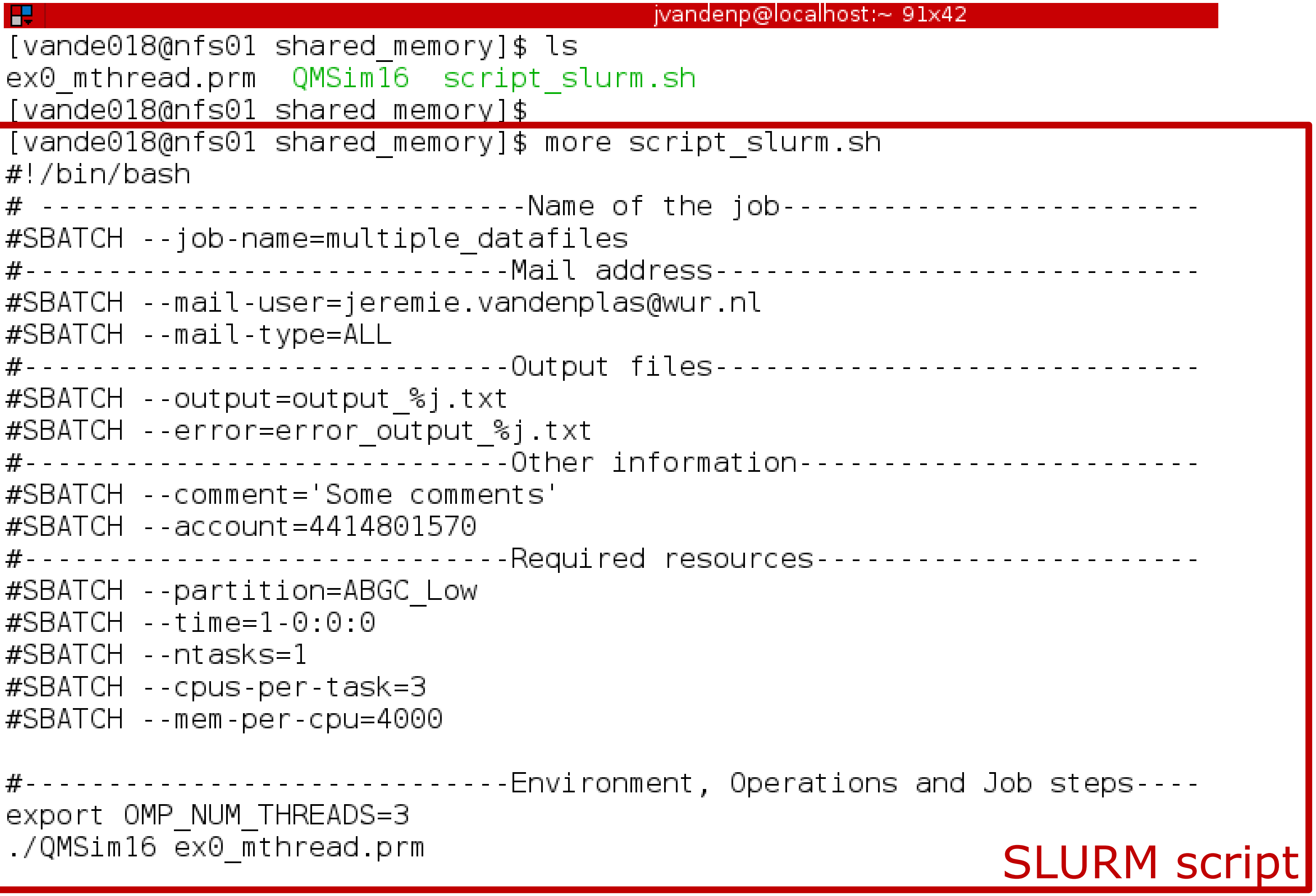
A shared memory example: resource

You want	SLURM options
To chose a partition	--partition=ABGC_Std
8 hours	--time=00-08:00:00
1 process that can use 3 cores for multithreading	--ntasks=1 --cpus-per-task=3
4000MB per CPU	--mem-per-cpu=4000
You use	export OMP_NUM_THREADS=3 (export MKL_NUM_THREADS=3) (srun) ./myprog

→ Run the job on a single node with

- max. 3 threads
- max. RAM = 3*4000=12000 MB

A shared memory example: script



jvandenp@localhost:~ 91x42

```
[vande018@nfs01 shared_memory]$ ls
ex0_mthread.prm  QMSim16  script_slurm.sh
[vande018@nfs01 shared_memory]$
[vande018@nfs01 shared_memory]$ more script_slurm.sh
#!/bin/bash
# -----------------------------Name of the job-------------------------
#SBATCH --job-name=multiple_datafiles
#-----------------------------Mail address-----------------------------
#SBATCH --mail-user=jeremie.vandenplas@wur.nl
#SBATCH --mail-type=ALL
#-----------------------------Output files-----------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=4414801570
#-----------------------------Required resources-----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=1-0:0:0
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=3
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
export OMP_NUM_THREADS=3
./QMSim16 ex0_mthread.prm
```

SLURM script

Pitfalls

- Using --ntasks=*n* for shared memory programs
  - Could work **or not**!
  - ➔Use --ntasks=1 --cpus-per-task=*n*

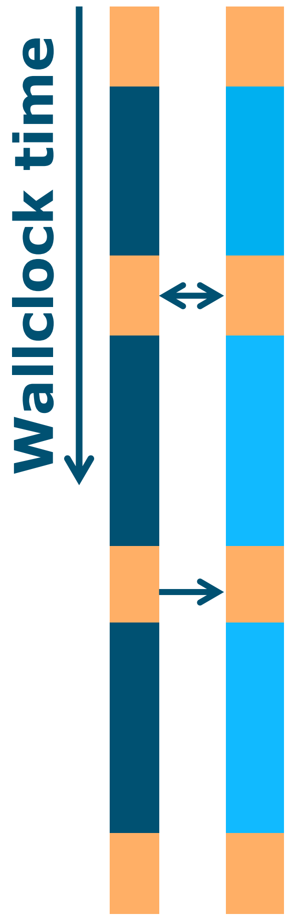
- Forgetting to mention the number of threads to the shared memory program (e.g., OpenMP programs)
  - ➔Add *export OMP_NUM_THREADS=1* to your *~/.bashrc*

Try it...

- Copy the following directory in your $HOME
  - /lustre/shared/training_slurm/shared_memory/training
- Write a Slurm script to run **calc_grm** with **3 threads** and a total of **12 GB RAM**
- Calc_grm is available in the module SHARED/calc_grm/main

- Other example
  - /lustre/shared/training_slurm/shared_memory/training1

A message passing example



- Parallelism is obtained by launching a multi-process program
    - E.g., using MPI
- One program spawns itself on several nodes
- Inter-process communication by the network

A message passing example: resource

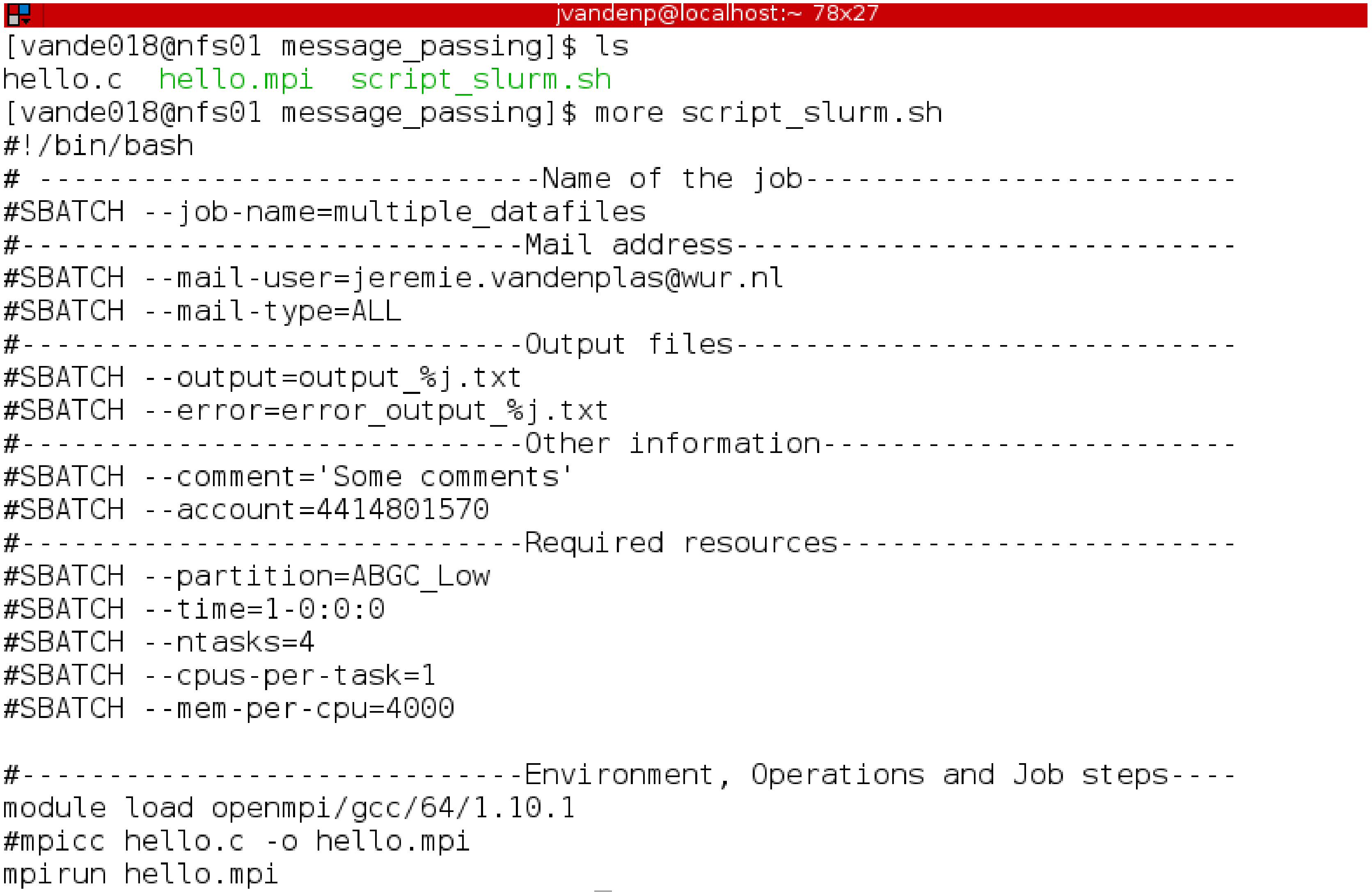
You want	SLURM options
To chose a partition	--partition=ABGC_Std
8hours	--time=00-08:00:00
3 processes for use with MPI that can use 1 core for multithreading	--ntasks=3 --cpus-per-task=1
4000MB per CPU	--mem-per-cpu=4000
You use	Module load mpi_library mpirun myprog

➔Run the job on max. 3 nodes with

- max. RAM = 3*4000=12000 MB

WAGENINGEN UR
For quality of life

A message passing example: script



jvandenp@localhost:~ 78x27

```
[vande018@nfs01 message_passing]$ ls
hello.c  hello.mpi  script_slurm.sh
[vande018@nfs01 message_passing]$ more script_slurm.sh
#!/bin/bash
# -----------------------------Name of the job-------------------------
#SBATCH --job-name=multiple_datafiles
#-----------------------------Mail address-----------------------------
#SBATCH --mail-user=jeremie.vandenplas@wur.nl
#SBATCH --mail-type=ALL
#-----------------------------Output files-----------------------------
#SBATCH --output=output_%j.txt
#SBATCH --error=error_output_%j.txt
#-----------------------------Other information------------------------
#SBATCH --comment='Some comments'
#SBATCH --account=4414801570
#-----------------------------Required resources-----------------------
#SBATCH --partition=ABGC_Low
#SBATCH --time=1-0:0:0
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

#-----------------------------Environment, Operations and Job steps----
module load openmpi/gcc/64/1.10.1
#mpicc hello.c -o hello.mpi
mpirun hello.mpi
```

WAGENINGEN UR
For quality of life

Pitfalls

- Using --ntaks=*n* for shared memory programs
  - Could work or not!
  - ➔Use --ntaks=1 --cpus-per-task=*n*
- Forgetting to mention the number of threads to the shared memory program
  - ➔Add *export OMP_NUM_THREADS=1* to your *~/.bashrc*
- Shared memory program OR message passing program?
  - ➔RTFM!
  - ➔Check the output of ***top*** with a small example!

WAGENINGEN UR
For quality of life

A mixed example

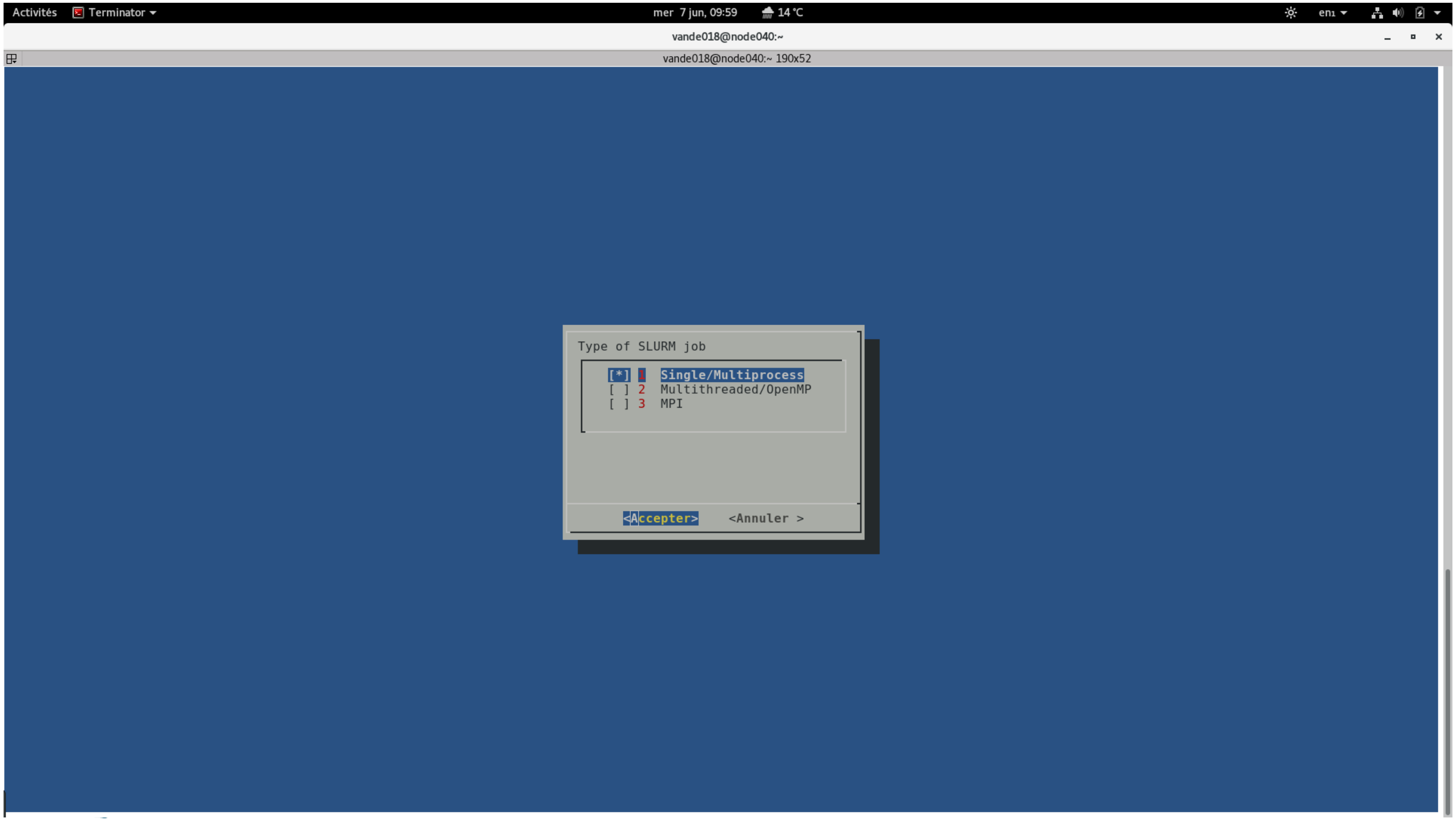
- A parallel job can included different parallelization paradigms!

You want	SLURM options
To chose a partition	--partition=ABGC_Std
8 hours	--time=00-08:00:00
4 processes that can use 3 cores for multithreading	--ntasks=4 --cpus-per-task=3
4000MB per CPU	--mem-per-cpu=4000
You use	Module load mpi_library export OMP_NUM_THREADS=3 (export MKL_NUM_THREADS=3) mpirun myprog

WAGENINGEN UR
For quality of life

Helpful tool

/cm/shared/apps/accounting/sbatch-generator



```
Type of SLURM job

  [*] 1   Single/Multiprocess
  [ ] 2   Multithreaded/OpenMP
  [ ] 3   MPI

        <Accepter>     <Annuler >
```

Summary: resource requests

- Choose the number of processes (--ntasks)
- Choose the number of threads per process (--cpu-per-task)

- Set environment variables (OMP_NUM_THREADS, MKL_NUM_THREADS,...)

- Use SLURM environment variables if required

- Launch processes with srun or mpirun if required

4. Monitoring and controlling a job

- Commonly used commands to monitor and control a job
  - ***squeue***
  - ***scontrol***
  - ***scancel***
  - ***sprio***

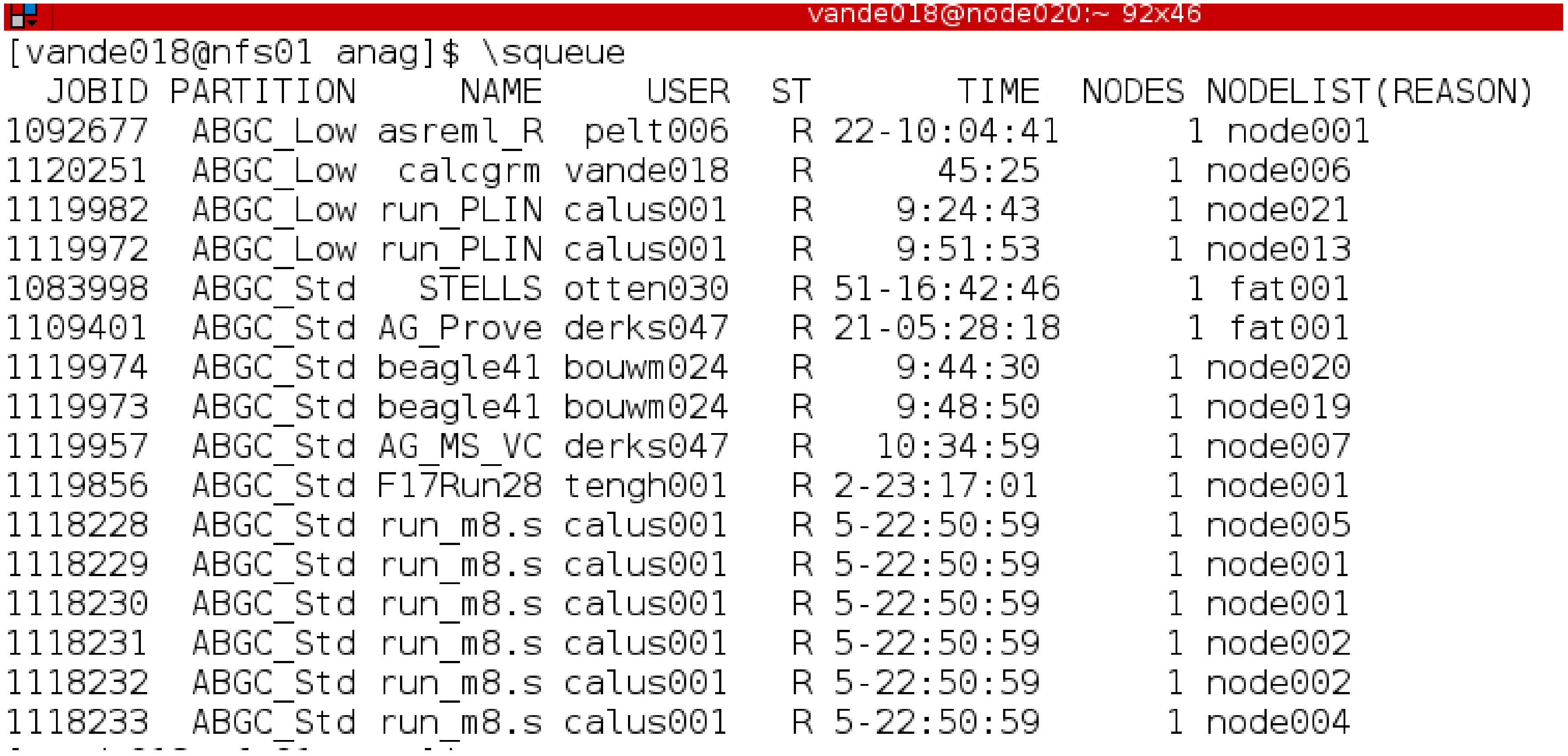
4. Monitoring and controlling a job
squeue

- ***squeue*** [options]
  - View information about jobs located in the SLURM scheduling queue
  - Useful options

Option	Report
-j <job_id_list>	Report for a list of **specific jobs**
-l	Report **time limit**
--start	Report the **expected start time** of pending jobs
-u <user_id_list>	Report for a list of **users**

4. Monitoring and controlling a job

squeue



```
vande018@node020:~ 92x46
[vande018@nfs01 anag]$ \squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
1092677   ABGC_Low asreml_R  pelt006   R 22-10:04:41      1 node001
1120251   ABGC_Low  calcgrm vande018   R      45:25      1 node006
1119982   ABGC_Low run_PLIN calus001   R    9:24:43      1 node021
1119972   ABGC_Low run_PLIN calus001   R    9:51:53      1 node013
1083998   ABGC_Std   STELLS otten030   R 51-16:42:46      1 fat001
1109401   ABGC_Std AG_Prove derks047   R 21-05:28:18      1 fat001
1119974   ABGC_Std beagle41 bouwm024   R    9:44:30      1 node020
1119973   ABGC_Std beagle41 bouwm024   R    9:48:50      1 node019
1119957   ABGC_Std AG_MS_VC derks047   R   10:34:59      1 node007
1119856   ABGC_Std F17Run28 tengh001   R 2-23:17:01      1 node001
1118228   ABGC_Std run_m8.s calus001   R 5-22:50:59      1 node005
1118229   ABGC_Std run_m8.s calus001   R 5-22:50:59      1 node001
1118230   ABGC_Std run_m8.s calus001   R 5-22:50:59      1 node001
1118231   ABGC_Std run_m8.s calus001   R 5-22:50:59      1 node002
1118232   ABGC_Std run_m8.s calus001   R 5-22:50:59      1 node002
1118233   ABGC_Std run_m8.s calus001   R 5-22:50:59      1 node004
```

WAGENINGEN UR
For quality of life

4. Monitoring and controlling a job *scontrol*

- ***scontrol*** [options] [command]
  - View Slurm configuration and state
  - Update job resource request
  - Work only for running jobs

  - Useful option

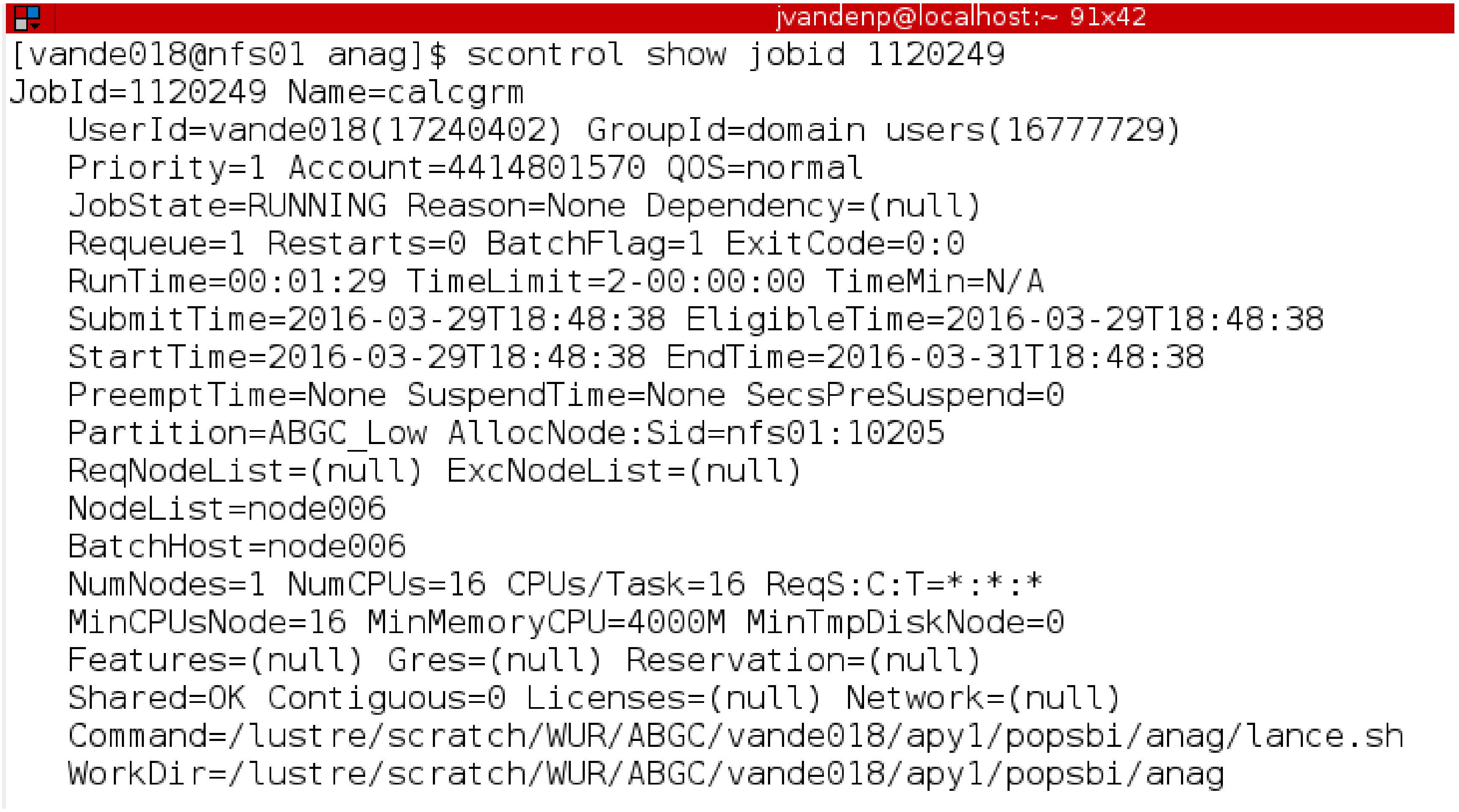
    scontrol show job *JOB_ID*

  ➔Lots of information

WAGENINGEN UR
For quality of life

4. Monitoring and controlling a job

scontrol



```
[vande018@nfs01 anag]$ scontrol show jobid 1120249
JobId=1120249 Name=calcgrm
   UserId=vande018(17240402) GroupId=domain users(16777729)
   Priority=1 Account=4414801570 QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 ExitCode=0:0
   RunTime=00:01:29 TimeLimit=2-00:00:00 TimeMin=N/A
   SubmitTime=2016-03-29T18:48:38 EligibleTime=2016-03-29T18:48:38
   StartTime=2016-03-29T18:48:38 EndTime=2016-03-31T18:48:38
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition=ABGC_Low AllocNode:Sid=nfs01:10205
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=node006
   BatchHost=node006
   NumNodes=1 NumCPUs=16 CPUs/Task=16 ReqS:C:T=*:*:*
   MinCPUsNode=16 MinMemoryCPU=4000M MinTmpDiskNode=0
   Features=(null) Gres=(null) Reservation=(null)
   Shared=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/lustre/scratch/WUR/ABGC/vande018/apy1/popsbi/anag/lance.sh
   WorkDir=/lustre/scratch/WUR/ABGC/vande018/apy1/popsbi/anag
```

WAGENINGEN UR
For quality of life

4. Monitoring and controlling a job
scancel

- ***scancel*** [options] [job_id[.step_id]...]
    - Cancel jobs or job steps

4. Monitoring and controlling a job
sprio

- ***sprio*** [options]
  - View the components of a job's scheduling priority
  - Rule: a job with a lower priority can start before a job with a higher priority IF it does not delay that jobs's start time
  - Useful options

Option	Report
-j <job_id_list>	Report for a list of **specific jobs**
-l	Report **more information**
-u <user_id_list>	Report for a list of **users**

WAGENINGEN UR
For quality of life

5. Getting an overview of jobs

- Previous and running jobs
    - ***sacct***
- Running jobs
    - ***scontrol***
    - ***sstat***
- *Previous jobs*
    - *Contents of emails (--mail-type=END|ALL)*

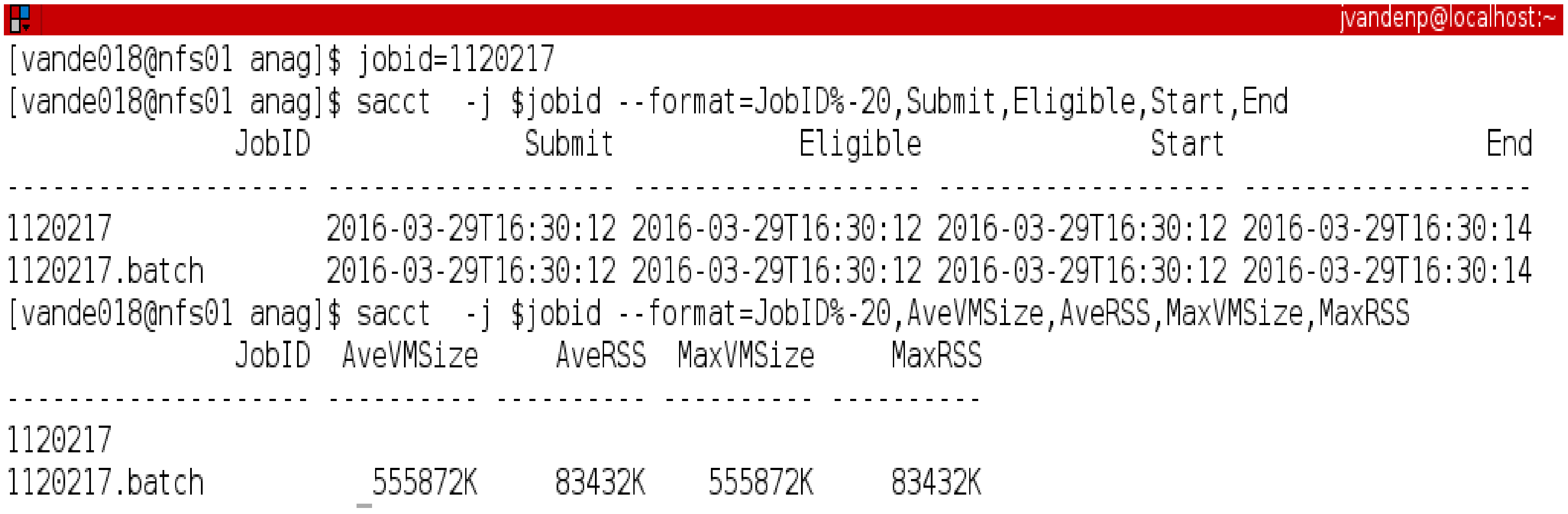
5. Getting an overview of jobs
sacct

- ***sacct*** [options]
  - Display accounting data for all jobs/steps
  - Some information are available only at the end of the job
  - Useful options

Option	Report
-j <job_id_list>	Report for a list of **specific jobs**
--format	Comma separated **list of fields**

5. Getting an overview of jobs

sacct



jvandenp@localhost:~

```
[vande018@nfs01 anag]$ jobid=1120217
[vande018@nfs01 anag]$ sacct  -j $jobid --format=JobID%-20,Submit,Eligible,Start,End
               JobID              Submit            Eligible               Start                 End
-------------------- ------------------- ------------------- ------------------- -------------------
1120217              2016-03-29T16:30:12 2016-03-29T16:30:12 2016-03-29T16:30:12 2016-03-29T16:30:14
1120217.batch        2016-03-29T16:30:12 2016-03-29T16:30:12 2016-03-29T16:30:12 2016-03-29T16:30:14
[vande018@nfs01 anag]$ sacct  -j $jobid --format=JobID%-20,AveVMSize,AveRSS,MaxVMSize,MaxRSS
               JobID  AveVMSize     AveRSS  MaxVMSize     MaxRSS
-------------------- ---------- ---------- ---------- ----------
1120217
1120217.batch           555872K     83432K    555872K     83432K
```

5. Getting an overview of running jobs *sstat*

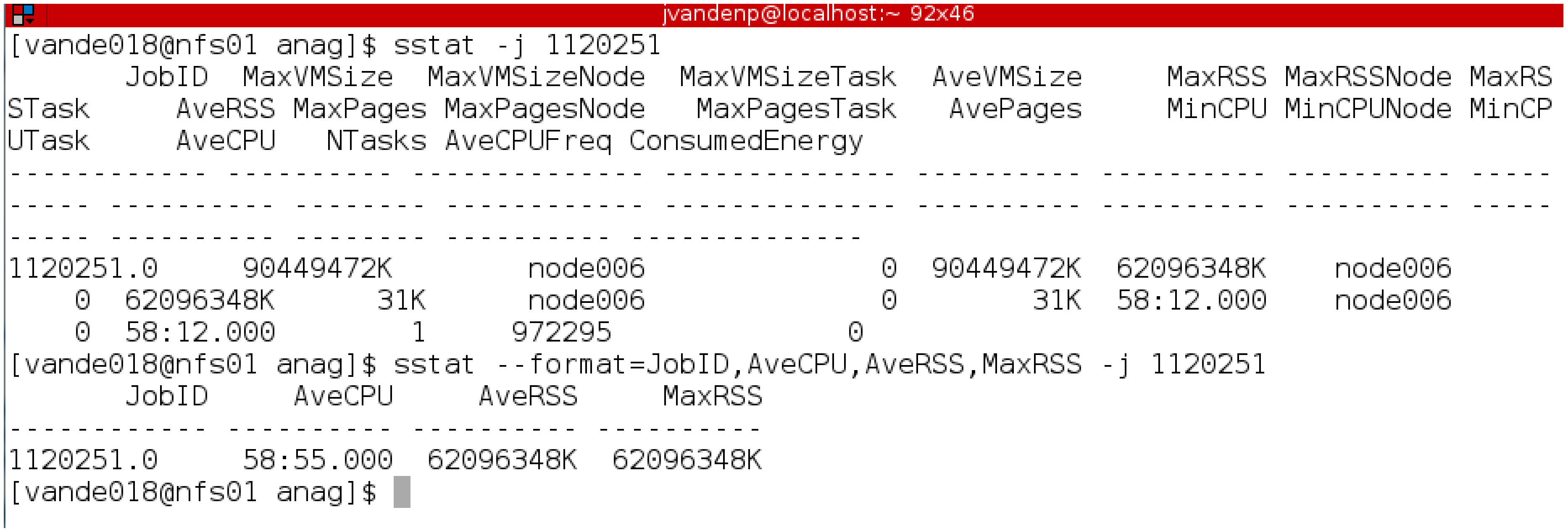
- ***sstat*** [options]
  - Display various status information of a running job/step
  - Work only if srun if used
  - Useful options

Option	Report
-j <job_id_list>	Report for a list of **specific jobs**
--format	Comma separated **list of fields**

WAGENINGEN UR For quality of life

5. Getting an overview of running jobs

sstat



jvandenp@localhost:~ 92x46

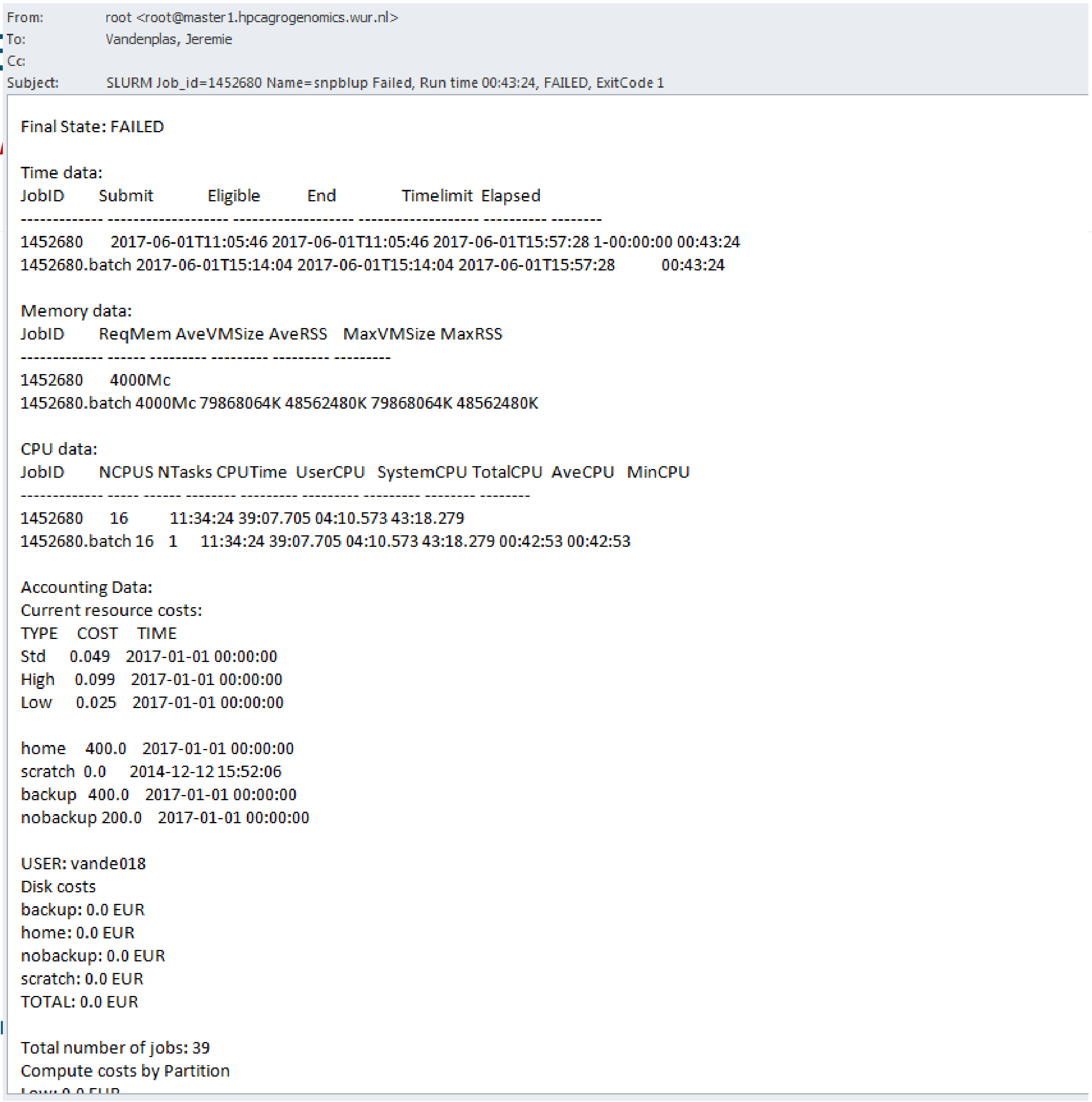
```
[vande018@nfs01 anag]$ sstat -j 1120251
       JobID  MaxVMSize  MaxVMSizeNode  MaxVMSizeTask  AveVMSize     MaxRSS MaxRSSNode MaxRS
STask     AveRSS MaxPages MaxPagesNode   MaxPagesTask   AvePages     MinCPU MinCPUNode MinCP
UTask     AveCPU   NTasks AveCPUFreq ConsumedEnergy
------------ ---------- -------------- -------------- ---------- ---------- ---------- -----
----- ---------- -------- ------------ -------------- ---------- ---------- ---------- -----
----- ---------- -------- ---------- --------------
1120251.0      90449472K        node006              0  90449472K  62096348K    node006
    0  62096348K      31K       node006              0        31K  58:12.000    node006
    0  58:12.000        1     972295              0
[vande018@nfs01 anag]$ sstat --format=JobID,AveCPU,AveRSS,MaxRSS -j 1120251
       JobID     AveCPU     AveRSS     MaxRSS
------------ ---------- ---------- ----------
1120251.0      58:55.000  62096348K  62096348K
[vande018@nfs01 anag]$ 
```

5. Getting an overview of jobs
emails

- Displays time, memory and CPU data

5. Ge
emai

- Displ

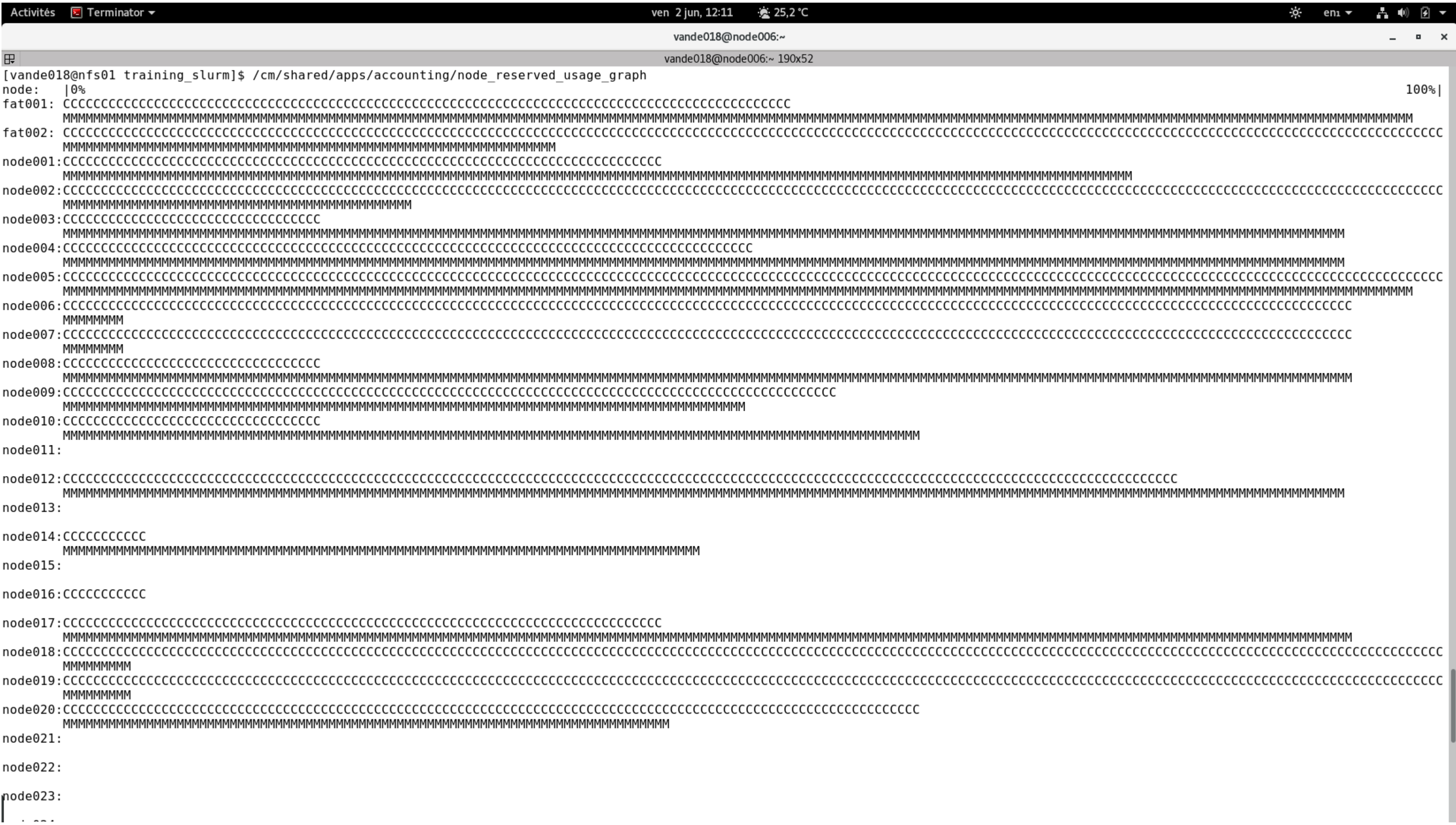


```
From:      root <root@master1.hpcagrogenomics.wur.nl>
To:        Vandenplas, Jeremie
Cc:
Subject:   SLURM Job_id=1452680 Name=snpblup Failed, Run time 00:43:24, FAILED, ExitCode 1

Final State: FAILED

Time data:
JobID        Submit              Eligible            End                 Timelimit  Elapsed
------------ ------------------- ------------------- ------------------- ---------- --------
1452680      2017-06-01T11:05:46 2017-06-01T11:05:46 2017-06-01T15:57:28 1-00:00:00 00:43:24
1452680.batch 2017-06-01T15:14:04 2017-06-01T15:14:04 2017-06-01T15:57:28           00:43:24

Memory data:
JobID        ReqMem AveVMSize AveRSS    MaxVMSize MaxRSS
------------ ------ --------- --------- --------- ---------
1452680      4000Mc
1452680.batch 4000Mc 79868064K 48562480K 79868064K 48562480K

CPU data:
JobID        NCPUS NTasks CPUTime  UserCPU   SystemCPU TotalCPU  AveCPU   MinCPU
------------ ----- ------ -------- --------- --------- --------- -------- --------
1452680      16           11:34:24 39:07.705 04:10.573 43:18.279
1452680.batch 16   1      11:34:24 39:07.705 04:10.573 43:18.279 00:42:53 00:42:53

Accounting Data:
Current resource costs:
TYPE    COST    TIME
Std     0.049   2017-01-01 00:00:00
High    0.099   2017-01-01 00:00:00
Low     0.025   2017-01-01 00:00:00

home    400.0   2017-01-01 00:00:00
scratch 0.0     2014-12-12 15:52:06
backup  400.0   2017-01-01 00:00:00
nobackup 200.0   2017-01-01 00:00:00

USER: vande018
Disk costs
backup: 0.0 EUR
home: 0.0 EUR
nobackup: 0.0 EUR
scratch: 0.0 EUR
TOTAL: 0.0 EUR

Total number of jobs: 39
Compute costs by Partition
Low: 0.0 EUR
```

WAG

Information on the HPC

/cm/shared/apps/accounting/node_reserve_usage_graph



```
[vande018@nfs01 training_slurm]$ /cm/shared/apps/accounting/node_reserved_usage_graph
node:   |0%                                                                                                                                                                                 100%|
fat001: CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
fat002: CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node001:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node002:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node003:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node004:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node005:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node006:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMM
node007:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMM
node008:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node009:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node010:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node011:

node012:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node013:

node014:CCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node015:

node016:CCCCCCCCCCC

node017:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node018:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMM
node019:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMM
node020:CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
node021:

node022:

node023:
```

Information on the HPC

- ***/cm/shared/apps/accounting/node_reserve_usage_graph***
- ***/cm/shared/apps/accounting/get_my_bill***
- ***sinfo***
- ***scontrol show nodes***

- **https://wiki.hpcagrogenomics.wur.nl/index.php/Log_in_to_B4F_cluster**

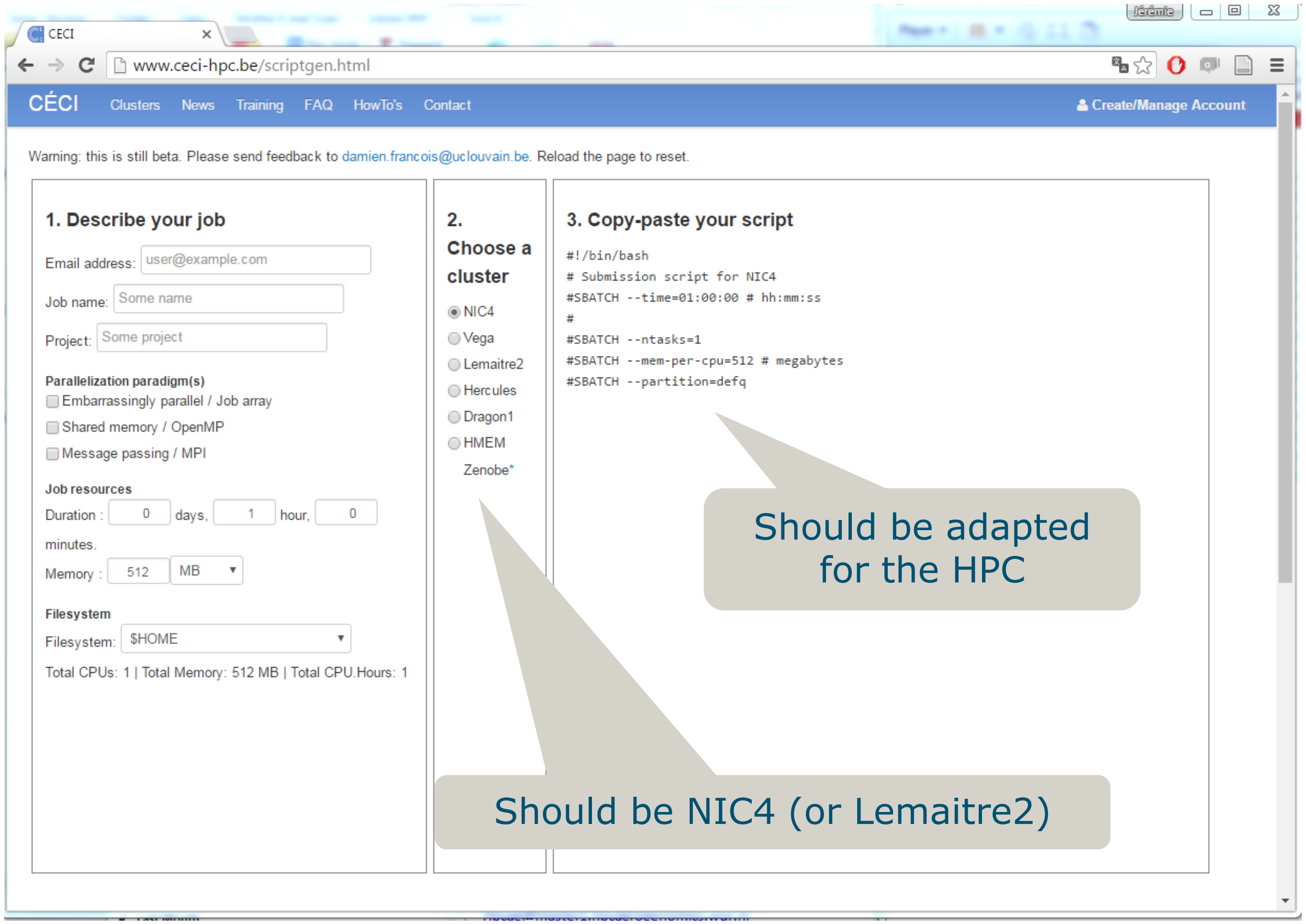
Thank you!

Questions?

WAGENINGEN UR
For quality of life

Helpful tool

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



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

1. Describe your job

Email address: user@example.com

Job name: Some name

Project: Some project

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 Zenobe*

3. Copy-paste your script

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

Should be adapted for the HPC

Should be NIC4 (or Lemaitre2)

WAGENINGEN UR For quality of life