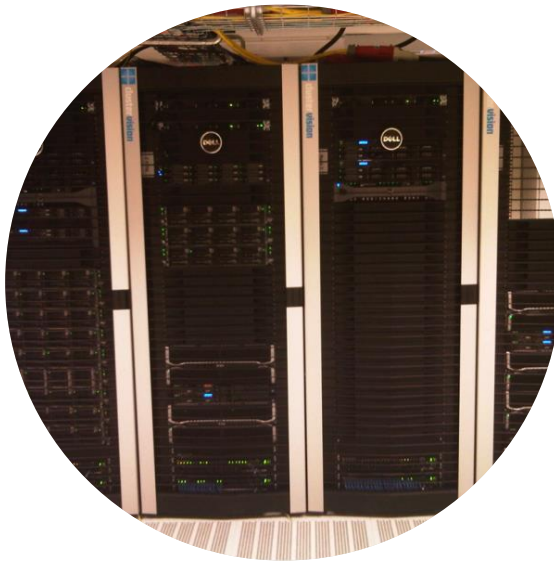


High Performance Computing Cluster

Basic course

Jeremie Vandenplas, Gwen Dawes

14 November 2019



Outline

- Introduction to the HPC Anunna
- Some “advanced” tools in Unix/Linux
- Submitting and monitoring basic jobs on Anunna

Introduction to the HPC Anunna

Jeremie Vandenplas, Gwen Dawes



Outline

- Some definitions
- Description of the HPC Anunna

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

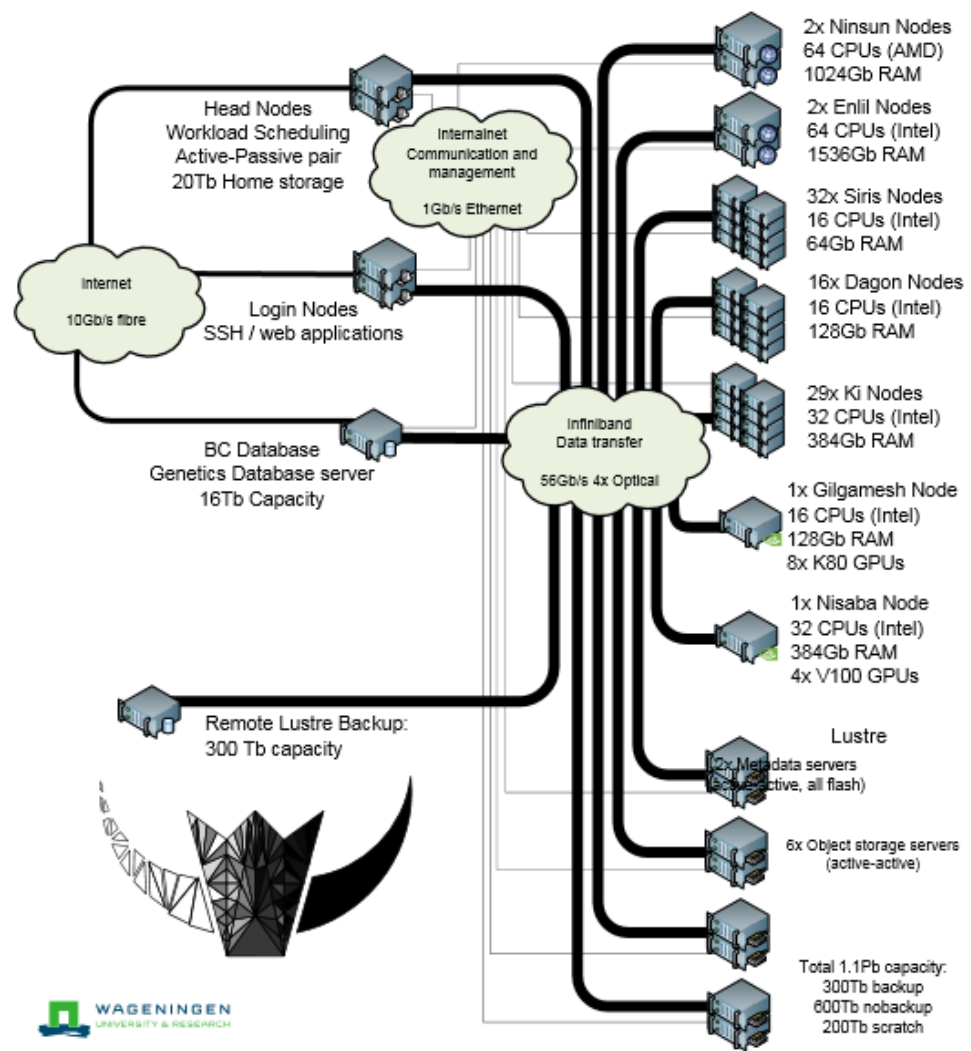
- High performance computing **cluster**
 - Group of **interconnected computers (node)** that work together and act like a single system
- **CPU** (Central processing unit)
 - Component within a computer that **carries out the instructions** of a computer program
- **Core**
 - **Processing unit** which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



ANUNNA

HIGH PERFORMANCE CLUSTER

Total compute capacity:
2000+ CPUs
19+ Tb RAM



HPC Anunna

- 48 Computes nodes
 - 16 cores (Intel), 64 GB or 128 GB RAM
- 29 Computes nodes
 - 32 cores (Intel), 328 GB RAM
- 2 Fat nodes
 - 64 cores (AMD), 1 TB RAM
- 2 Fat nodes
 - 64 cores (Intel), 1.5 TB RAM
- 2 GPU nodes
 - NVIDIA
- 1000 TB Lustre parallel file system (15 GB/s)

HPC Anunna – main storage

■ Home directory

- /home/[partner]/[username]
- Directory where you are after logon
- Quota of 200GB soft (210GB hard)

■ Archive

- /archive/[partner]/[username]
- Cheap
- **Only** for **storage** and for **WUR**

HPC Anunna – main storage

- **Lustre** filesystem (faster storage)
 - **backup**
 - /lustre/backup/[partner]/[unit]/[username]
 - Extra cost for backup
 - **nobackup**
 - /lustre/nobackup/[partner]/[unit]/[username]
 - Some costs
 - **scratch**
 - /lustre/scratch/[partner]/[unit]/[username]
 - Free
 - (Regularly cleaned up)

HPC Anunna – “rules”

■ Home

- Jobscripts
- Small datasets (performance)
- Not computational jobs

■ Lustre

- Big datasets
- Intensive (computing) jobs
- No job run outside SLURM

■ Archive

- No job

HPC Anunna – useful information

- HPC Anunna wiki
 - https://wiki.anunna.wur.nl/index.php/Main_Page
- Linux User Group at WUR
 - https://lug.wur.nl/index.php/Main_Page
- Support
 - hpc.support@wur.nl

Questions?

Some “advanced” tools in Unix/Linux

Jeremie Vandenplas, Gwen Dawes



(De)compressing files

- To compress a file:

gzip file1

- To decompress a file:

gunzip file1.gz

gzip -d file2.gz

- Other commands

bzip2, xz, zip,...

Transferring files using *scp*

- To copy a file *from* an external machine:

scp username@hostname:~/file1 destination_name

- To copy a file *to* an external machine:

scp ~/file1 username@hostname:destination_name

Downloading files from the web

- To download a file from the web:

wget [options] [url]

Making a file executable

- To make a file executable

chmod u+x file1

- To execute a program/script/....

./program [options]

/path/to/the/program/program [options]

Environment variables

- ~data storage for Unix/Linux shell

- To assign an environment variable

MYVARIABLE=my_value

- To access the data stored within an environment variable:

echo \$MYVARIABLE

- To list all environment variables:

env

- Remove the existence of an environment variable:

unset MYVARIABLE

Environment modules

- Provides many software not installed by default
- module avail
- module list
- module load name
- module rm

A bash (Shell) script

- **Plain text file** which contains a serie/mixture of **commands**.
- **Tip**
 - **Anything** you can run normally on the **command line** can be put **into a script** and it will do exactly the same thing.
- Convention: extension of **.sh** (e.g., script.sh).
- Example

nfs01.hpcagrogenomics.wur.nl - PuTTY

```
9 #!/bin/bash ← Shebang with path of interpreter
8
7 #Create the directory ← Comment
6 mkdir serial_example ← Command
5
4 #go in the created directory
3 cd serial_example
```

Try it...

1. Create a directory (e.g., 'example_1') in your Lustre scratch directory
 2. Download QMSim from this URL (**wget**) and decompress (**unzip**) it:

<https://git.wur.nl/dawes001/public-files/raw/master/QMSim-Linux.zip>
 3. Copy the parameter file
`/lustre/shared/training_slurm/autumn_2019/serial/training/ex_serial_qmsim.prm`
in your directory!
- Extra: write a bash script to do all these steps!

Questions?

Solution

```
1  #!/bin/bash
2  #Environment variable
3  namedir='example_1'
4
5  #delete the directory if it exists
6  if [ -d $namedir ]; then rm -r $namedir; fi
7
8  #Create the directory
9  mkdir $namedir
10
11 #go in the created directory
12 cd $namedir
13
14 #download the archive QMSim
15 wget https://git.wur.nl/dawes001/public-files/raw/master/QMSim-Linux.zip
16
17 #decompress the archive
18 unzip QMSim-Linux.zip
19
20 #copy the parameter file
21 cp /lustre/shared/training_slurm/ autumn_2019 /serial/training/ex_serial_qmsim.prm .
22
23 #list the content of the directory
24 ls
```

Extra - Symbolic link

- To create a symbolic link to a file/directory, instead of copying it:

ln -s /path/to/file1 link

Submitting and monitoring basic jobs on Anunna

J. Vandenplas, G. Dawes



Outline

- Some definitions
- **Running a basic job on the nodes of Anunna**
 - Introduction to SLURM
 - Characteristics of a job
 - Writing and submitting a script
 - Monitoring and controlling a job
- Some exercises
- (Extra: Submitting a job array)

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



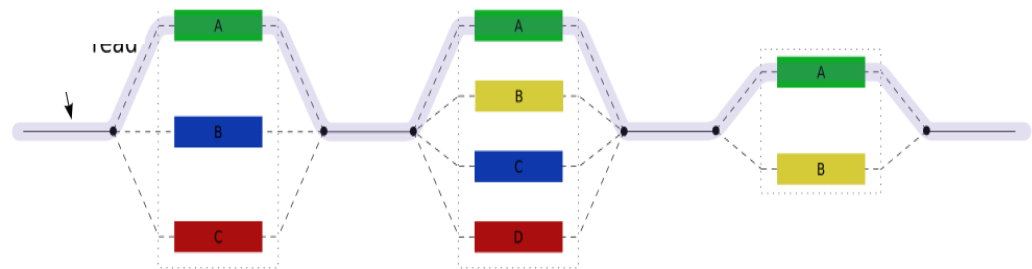
Some definitions

■ 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	0.0	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

Running a job on the nodes of Anunna?

Running a job on the nodes of Anunna?

■ 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 Anunna?

■ 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** (compute 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 basic job on the HPC nodes?

- 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'

#-----Required resources-----

#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?

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***

1. Characteristics of the job

- Try to **fit** to the **real use** as much as possible!
- Try to ask
 - 4 GB RAM per CPU for nodes with 64 GB
 - 8 GB RAM per CPU for nodes with 128 GB
 - 10.2 GB RAM per CPU for nodes with 328 GB
 - 15.6 GB RAM per CPU for nodes with 1 TB
 - 23.4 GB RAM per CPU for nodes with 1.5 TB

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 -c <cpus> --mem <MB>

1. Characteristics of the job

vande018@login0:~

```
[vande018@login0 ~]$ hostname  
login0
```

```
[vande018@login0 ~]$ sinteractive -c 1 --mem 2000  
srun: job 19271078 queued and waiting for resources  
srun: job 19271078 has been allocated resources  
[vande018@node018 ~]$ █
```

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

Try it...

1. Create a directory (e.g., 'example_1') in your Lustre scratch directory
2. Download QMSim from this URL and decompress it:
<https://git.wur.nl/dawes001/public-files/raw/master/QMSim-Linux.zip>
3. Copy the parameter file
/lustre/shared/training_slurm/spring_2018/serial/training/ex_serial_qmsim.prm
in your directory!
4. Try to find the **requirements** (e.g., memory) of **QMSim16** using **sinteractive**

(The parameter file must be mentioned in the command line)

Questions?

2. Writing a submission script

```
#!/bin/bash
# -----Name of the job-----
#SBATCH --job-name=serial_example
#-----Mail address-----
#SBATCH --mail-user=my.email.address@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'
#-----Required resources-----

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

echo 'Hello from a single task'
srun echo 'Hello from EACH task'
```

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

The Slurm command *srun*

```
[vande018@nfs01 vande018]# cat script_slurm.sh
[van018@nfs01 vande018]$ cat script_slurm.sh
#!/bin/bash
# -----Name of the job-----
#SBATCH --job-name=serial_example
#-----Mail address-----
#SBATCH --mail-user=my.email.address@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'
#-----Required resources-----

#SBATCH --time=0-1
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

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

echo 'Hello from a single task'

srun echo 'Hello from EACH task'

[vande018@nfs01 vande018]$
[vande018@nfs01 vande018]$ cat output_10969988.txt
Hello from a single task
Hello from EACH task
Hello from EACH task
Hello from EACH task
Hello from EACH task
[vande018@nfs01 vande018]$
[vande018@nfs01 vande018]$
```

Some SLURM options

You want	SLURM option
To set a job name	<code>--job-name="job1"</code>
To get emails	<code>--mail-user=name.name@wur.nl</code> <code>--mail-type=BEGIN END FAILED ALL</code>
To set the name of the output files	<code>--output=output_%j.txt</code> <code>--error=error_output_%j.txt</code>
To attach a comment to the job	<code>--comment="abcd"</code>

Some SLURM options: resource

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

Some SLURM options: features

- 128g/384g/1019g/1536g/normalmem/largemem/moremem
 - Nodes with **specific RAM**
- 16cpus/32cpus/64cpus
 - Nodes with a **specific total number of CPUs**
- 4gpercpu/8gpercpu/16gpercpu/24gpercpu
- nvidia/K80/V100
 - Nodes with **GPUs**
- Amd/avx512/intel
 - Nodes with **specific processors**
- dagon/enlil/gilgamesh/ki/ninsun/siris/gen2

Some SLURM options: resource

You want	SLURM option
To choose a partition	--partition=main gpu <i>Default: main</i>
To choose a Quality of Service	--qos=low std high interactive <i>Default: std</i>

Some SLURM options: quality of service

- low
 - Limited time (8h)
 - Very cheap
- std
 - No limit
- high
 - No limit + extra costs
- interactive
 - Immediate running jobs
 - A few jobs only

3. Submitting a job

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

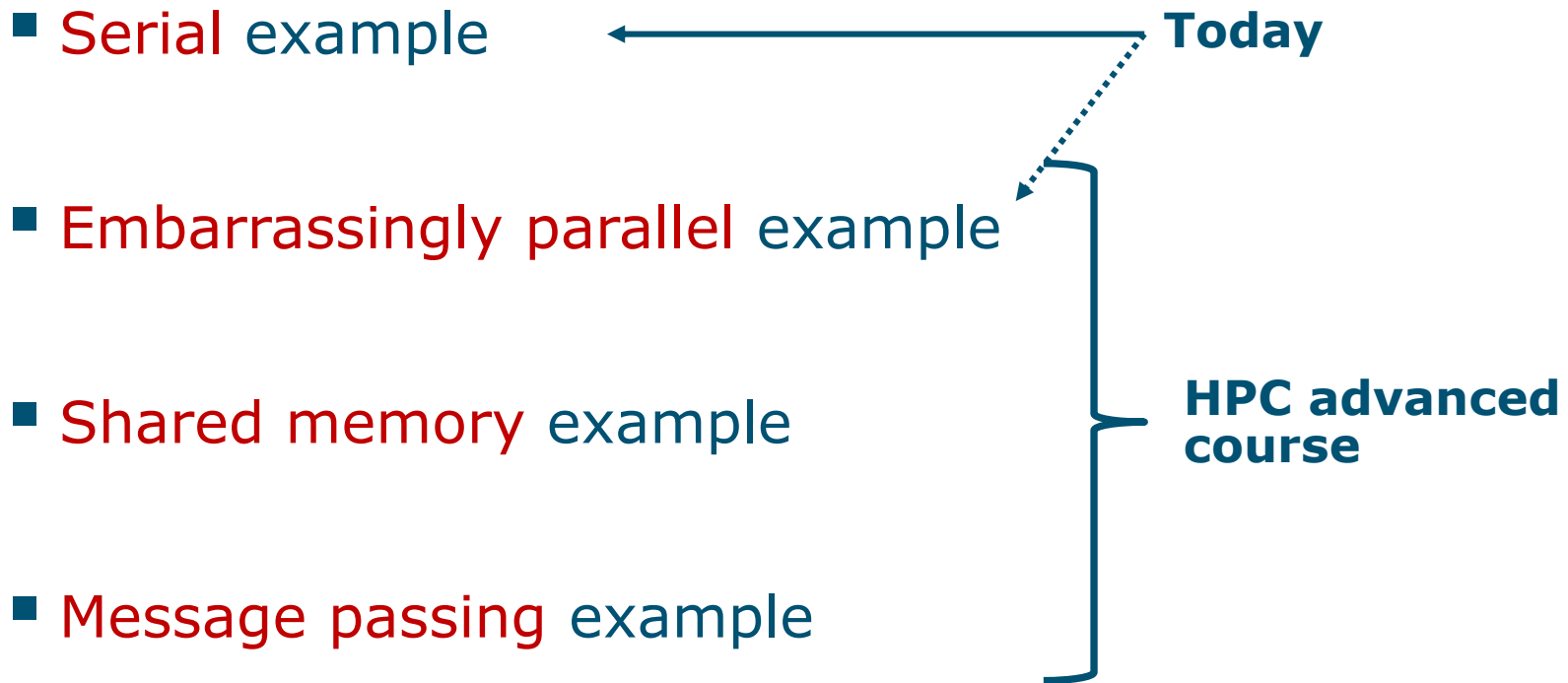
```
jvandenp@localhost:~ 91x42
[vande018@nfs01 shared_memory]$ ls
ex0_mthread.prm  QMSim16  script_slurm.sh
[vande018@nfs01 shared_memory]$
[vande018@nfs01 shared_memory]$ sbatch script_slurm.sh
Submitted batch job 1120242
```

- 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

Some jobs and their option requirements



A serial example



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

A **serial** example: resource

You want	SLURM options
8 hours	<code>--time=00-08:00:00</code>
1 independent process	<code>--ntasks=1</code>
4000MB per CPU	<code>--mem-per-cpu=4000</code>
You use	<code>(srun) ./myprog</code>

A serial example: script

```
#!/bin/bash
# -----Name of the job-----
#SBATCH --job-name=serial_example
#-----Mail address-----
#SBATCH --mail-user=my.email.address@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'
#-----Required resources-----
:
#SBATCH --time=0-1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

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

srun echo 'Hello'
```

4. Monitoring and controlling a job *scancel*

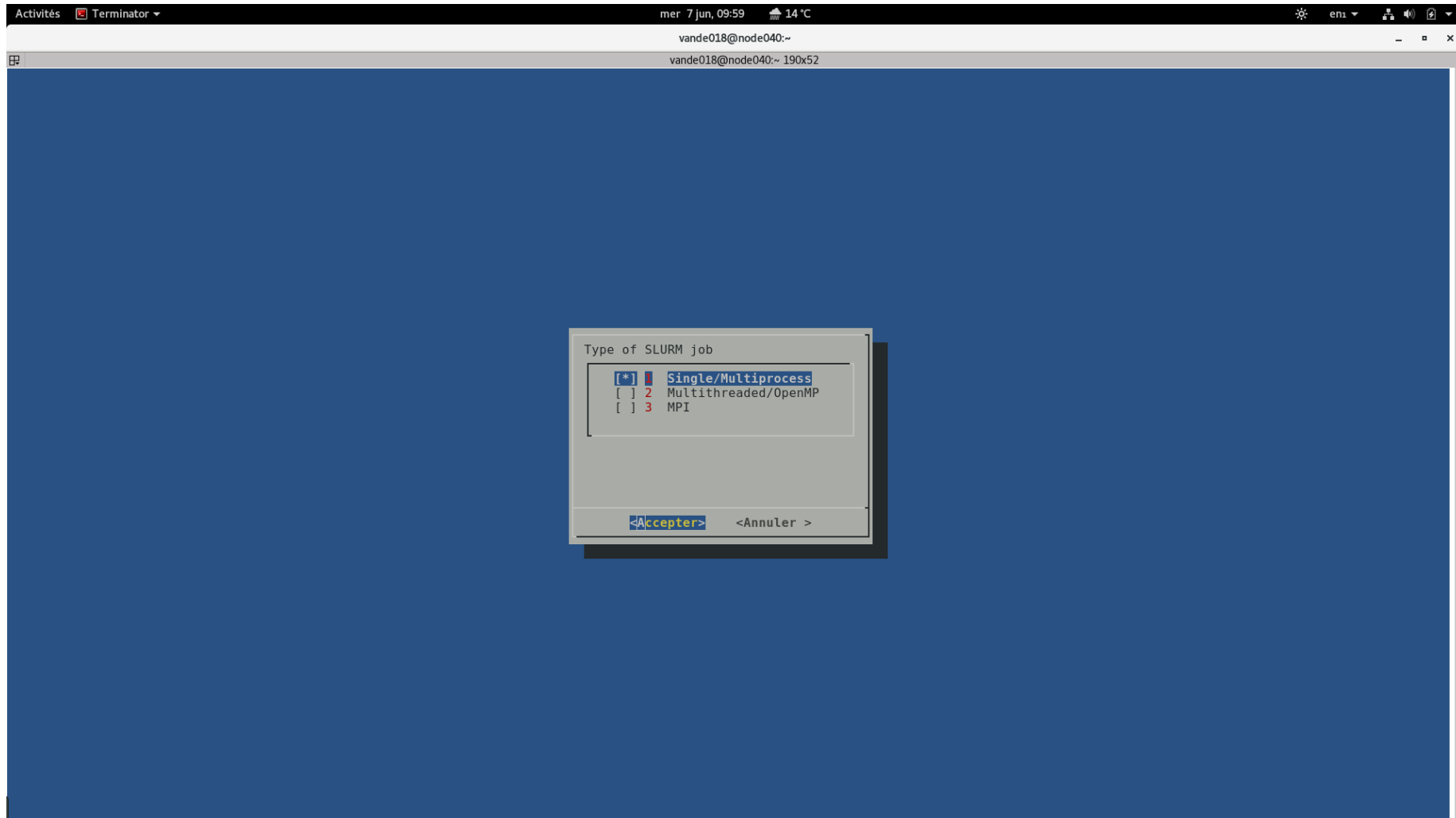
- *scancel* [options] [job_id[.step_id]...]
 - Cancel jobs or job steps

Try it...

- Write a Slurm script to run **QMSim16** with the required memory and submit it!

Helpful tool

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



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

queue

```
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
```

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 options

scontrol show job JOB_ID

scontrol show nodes

→ *Lots of information*

4. Monitoring and controlling a job

scontrol

```
jvandenp@localhost:~ 91x42
[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/apyl/popsbi/anag/lance.sh
  WorkDir=/lustre/scratch/WUR/ABGC/vande018/apyl/popsbi/anag
```

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 job'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

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

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

ema

■ Displ

From: root <root@master1.hpcagrogeomics.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

Information on the HPC Anunna

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

Extra information – job array

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
- It could be submit several times manually
- Or use job arrays!

An embarrassingly parallel example

Resource

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



```
[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'
```

```
#-----Required resources-----
```

```
#SBATCH --time=0-1
```

```
#SBATCH --array=1-3
```

```
#SBATCH --ntasks=1
```

```
#SBATCH --cpus-per-task=1
```

```
#SBATCH --mem-per-cpu=4000
```

3 array jobs
(from 1 to 3)

```
#-----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
```

SLURM script

```
[vande018@nfs01 one_parameter_file]$
```

Try it...

- Write a Slurm script to run **4 times** the program **QMSim16** with **1 thread** and a total of **4 GB RAM**.

Thank you!

Questions?



Helpful tool

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

The screenshot shows a web browser window displaying the CECI script generator. The page has a blue header with the CECI logo and navigation links (Clusters, News, Training, FAQ, HowTo's, Contact) and a 'Create/Manage Account' button. A warning message states: 'Warning: this is still beta. Please send feedback to damien.francois@uclouvain.be. Reload the page to reset.'

The form is divided into three main sections:

- 1. Describe your job:** Includes input fields for 'Email address' (user@example.com), 'Job name' (Some name), and 'Project' (Some project). It has checkboxes for 'Parallellization paradigm(s)' (Embarrassingly parallel / Job array, Shared memory / OpenMP, Message passing / MPI) and 'Job resources' (Duration: 0 days, 1 hour, 0 minutes; Memory: 512 MB). A 'Filesystem' dropdown is set to '\$HOME'. A summary line reads: 'Total CPUs: 1 | Total Memory: 512 MB | Total CPU.Hours: 1'.
- 2. Choose a cluster:** A list of radio buttons for cluster selection: NIC4 (selected), Vega, Lemaitre2, Hercules, Dragon1, HMEM, and Zenobe*.
- 3. Copy-paste your script:** A text area containing a shell 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
```

Two grey callout boxes with blue text provide feedback:

- A box pointing to the 'NIC4' radio button says: 'Should be NIC4 (or Lemaitre2)'.
- A box pointing to the script content says: 'Should be adapted for the HPC'.