High Performance Computing Cluster Advanced course

Jeremie Vandenplas, Gwen Dawes

16 October 2018

WAGENINGEN UR
For quality of life

Outline

- Introduction to the Agrogenomics HPC
- Submitting and monitoring jobs on the HPC
- Parallel jobs on the HPC
- Tips and tricks

Introduction to the Agrogenomics HPC

Jeremie Vandenplas, Gwen Dawes

WAGENINGEN UR
For quality of life

Agrogenomics HPC

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

➔Being upgraded!

Agrogenomics HPC – 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

Agrogenomics HPC – 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

Agrogenomics HPC – "rules"

- Home
  - Jobscripts
  - Small datasets (performance)
  - No computational jobs
- Lustre
  - Big datasets
  - Intensive (computing) jobs
  - No job run outside SLURM
- Archive
  - No job

Agrogenomics HPC – useful information

- Linux User Group at WUR
  - https://lug.wur.nl/index.php/Main_Page
- HPC wiki
  - https://wiki.hpcagrogenomics.wur.nl
- Contact person
  - Gwen Dawes
  - Jan van Lith

Questions?

Submitting and monitoring basic jobs on the HPC

J. Vandenplas, G. Dawes

WAGENINGEN UR
For quality of life

Outline

- Running a job on the nodes of the HPC
  - Introduction to SLURM
  - Characteristics of a job
  - Writing and submitting a script
  - Monitoring and controlling a job
  - Tips and tricks
- Types of jobs
  - Sequential
  - Array
  - Shared memory
  - Distributed memory

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

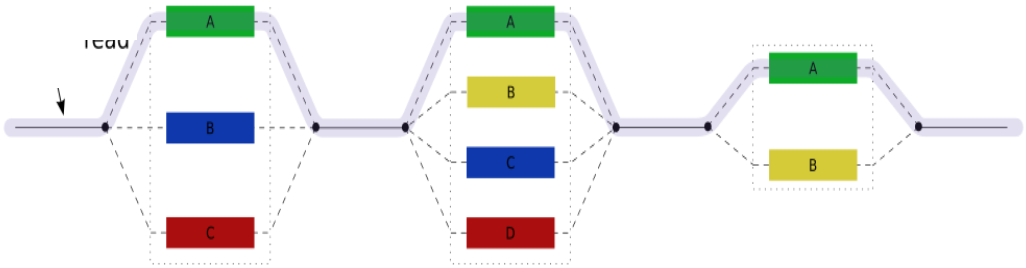
Job scheduler/Resource manager

- HPC's job scheduler: **SLURM** (Simple Linux Utility for Resource Management ; http://slurm.schedmd.com/slurm.html)

- 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

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



WAGENINGEN UR
For quality of life

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?

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

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

➔How can you know it?

  - RTFM!
  - Read the source code (if available)
  - Just run it!

    ➔use ***sinteractive***!

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)

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----------------------
#SBATCH --partition=ABGC_Std
#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'
```

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
#!/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 --partition=ABGC_Std
#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]$
```

Some SLURM options

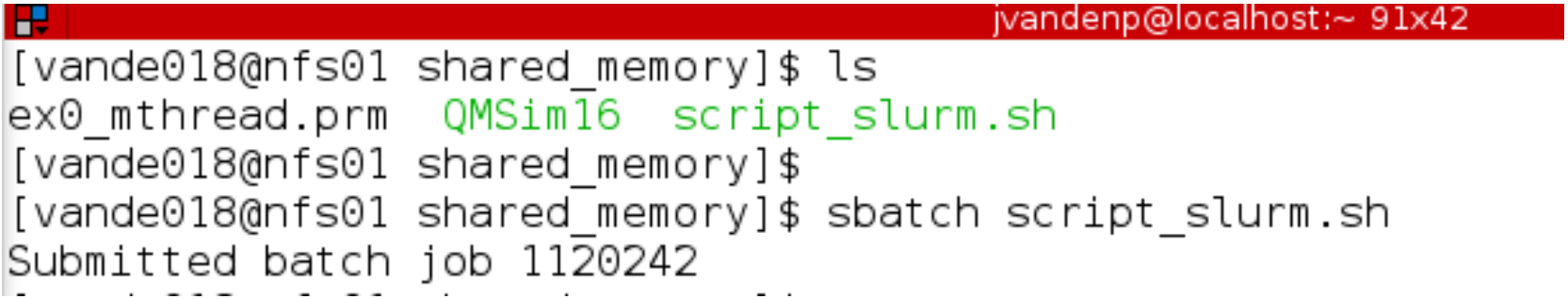
You want	SLURM option
To set a **job name**	--job-name="job1"
To get **emails**	--mail-user=name.name@wur.nl --mail-type=BEGIN|END|FAILED|ALL
To set the name of the **output files**	--output=output_%j.txt --error=error_output_%j.txt
To attach a **comment** to the job	--comment="abcd"

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
12 hours	--time=0-12:00:00
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



  - 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
  - See Gwen 's tips and tricks

4. Monitoring and controlling a job

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

- More details in Gwen's presentation

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
```

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**

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

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 is 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.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

Information on the HPC

/cm/shared/apps/accounting/node_reserve_usage_graph

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**

Gwen ‘s presentation

- Scontrol
- Sbatch
- Dependencies
- …

Parallel jobs on the HPC

Jeremie Vandenplas, Gwen Dawes

WAGENINGEN UR
For quality of life

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

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'

#------------------------------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

~
```

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
  - with some tricks (loops, srun environment variables,...)
- 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
6 processes to launch 6 completely independent jobs	--array=1,3-5,8
3 processes to launch 3 completely independent jobs (2 at a time)	--array=1-3%2
1 process per array	--ntasks=1
4000MB per CPU	--mem-per-cpu=4000
You use	$SLURM_ARRAY_TASK_ID (srun) ./myprog

```
[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 --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
```

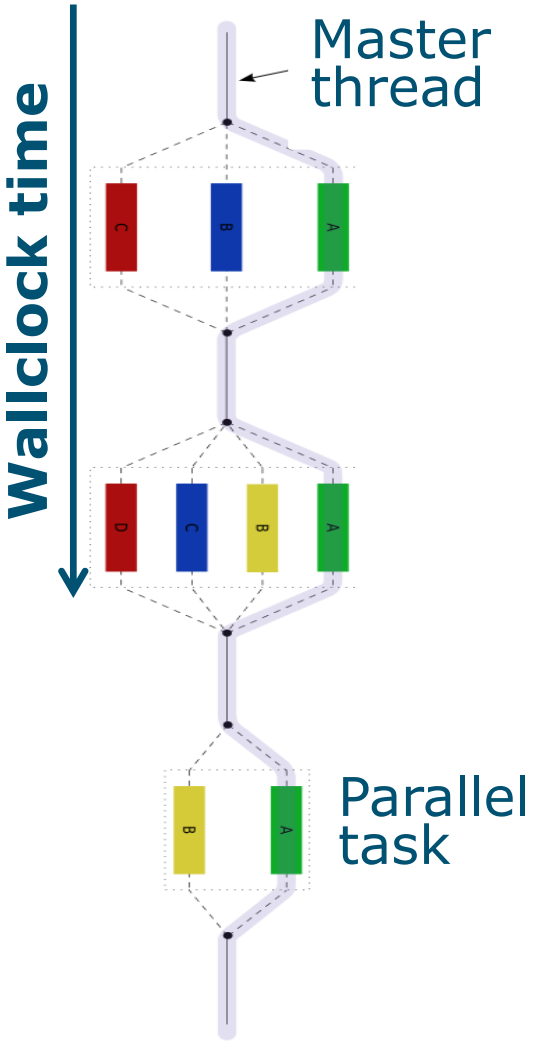
Useful: %A_%a

3 array jobs
(from 1 to 3)

SLURM script

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

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'

#------------------------------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

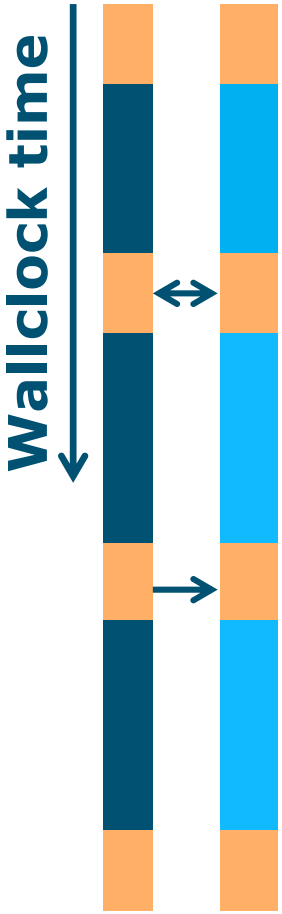
- Using --ntasks=*n* for shared memory programs
  - Could work **or not**!

  ➔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*

A message passing example



- Parallelism is obtained by launching a multi-process program
  - E.g., MPI, PGAS (Coarray Fortran, UPC)
- 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

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'

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

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

  ➔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!

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

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

Thank you!

Questions?

WAGENINGEN UR
For quality of life