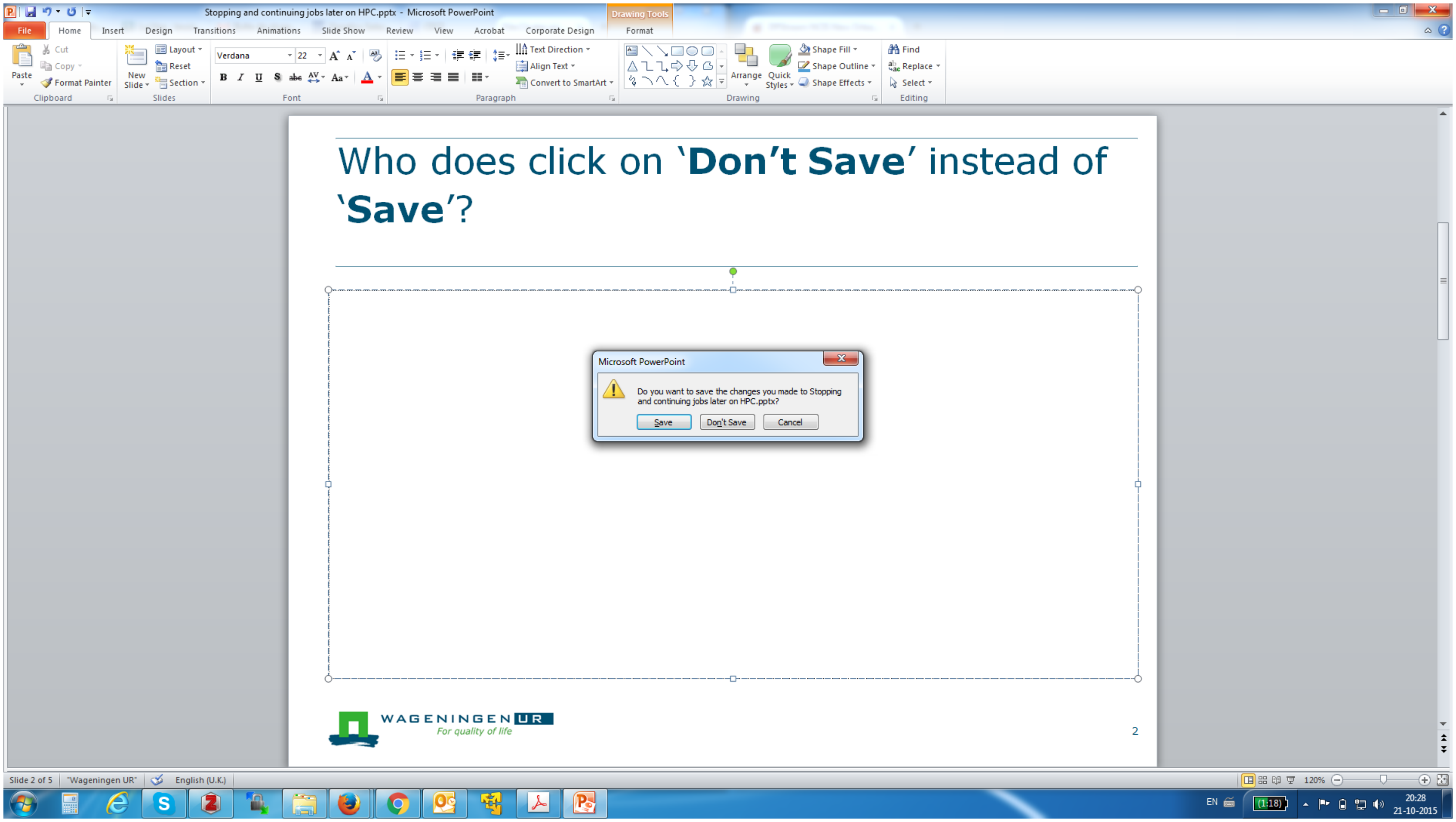
Checkpointing jobs on the HPC

Jérémie Vandenplas, Gwen Dawes

9 November 2017

WAGENINGEN UR
For quality of life

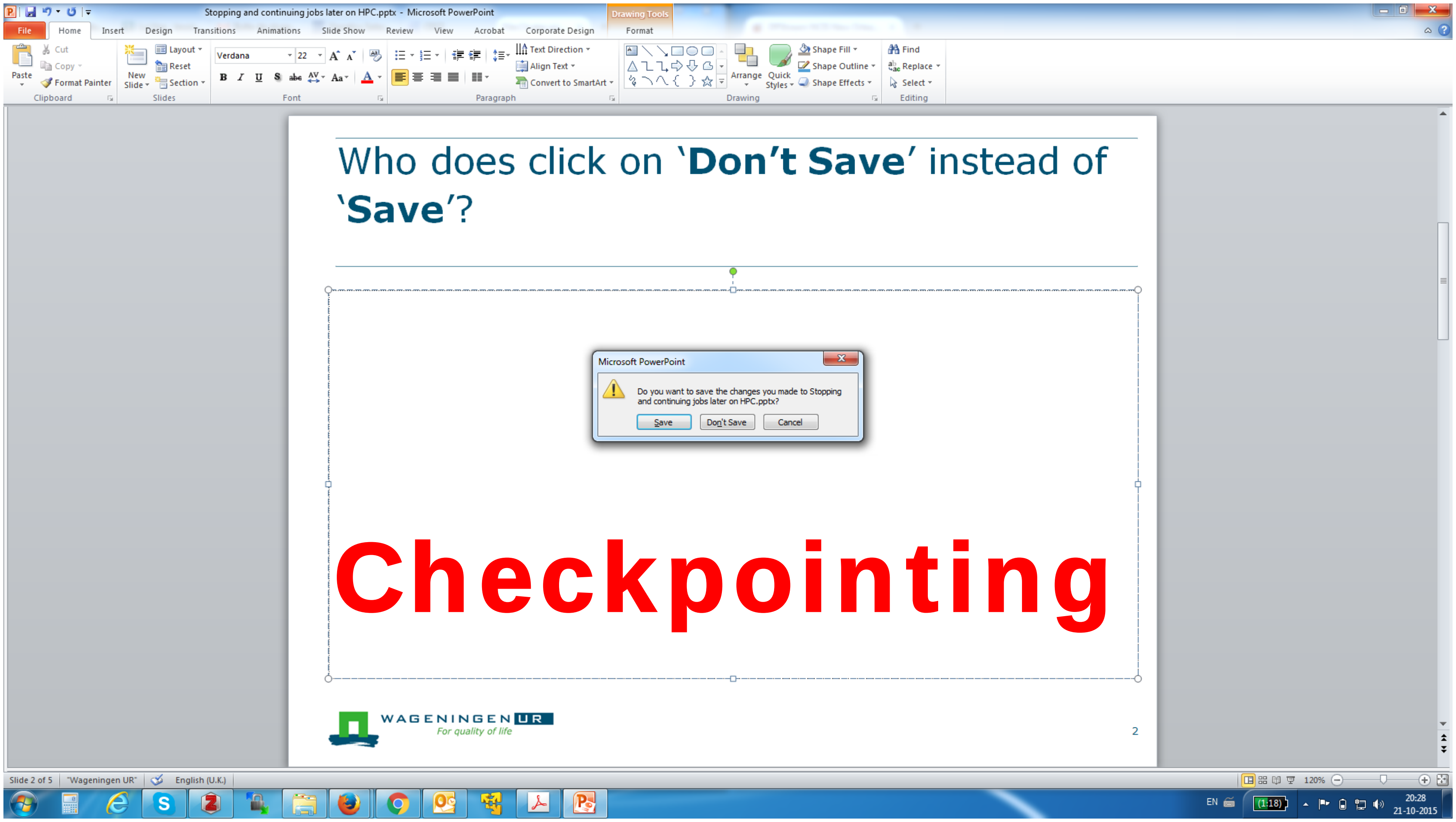
Who does click on ‘**Don’t Save**’ instead of ‘**Save**’?



Do you want to save the changes you made to Stopping and continuing jobs later on HPC.pptx?

Stopping and continuing jobs later on HPC.pptx - Microsoft PowerPoint
Microsoft PowerPoint
Save
Don't Save
Cancel

Who does click on ‘**Don’t Save**’ instead of ‘**Save**’?


Who does click on ‘**Don’t Save**’ instead of ‘**Save**’?

Microsoft PowerPoint

Do you want to save the changes you made to Stopping and continuing jobs later on HPC.pptx?

Save
Don't Save
Cancel
Stopping and continuing jobs later on HPC.pptx - Microsoft PowerPoint
WAGENINGEN UR
For quality of life
Slide 2 of 5

Checkpointing

Checkpointing

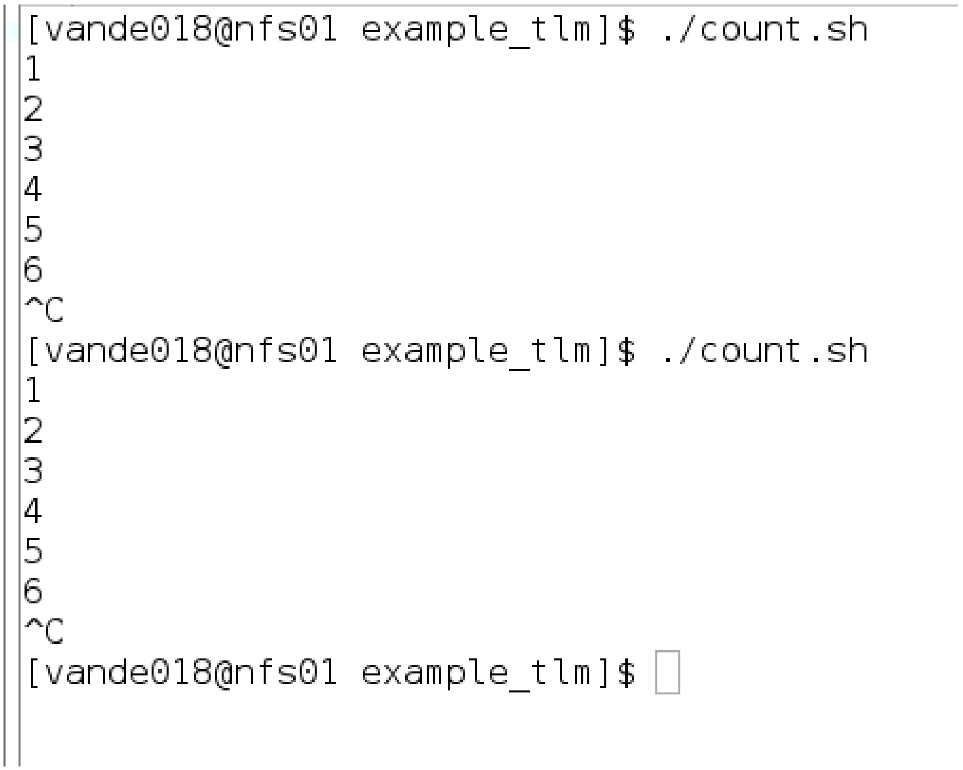
Saving the program's state
at a checkpoint
with the aim to restart it from that point
in case of (un)planned stop or failure

➔ Interesting for

- Long jobs that could be (un)voluntary killed
- Unstable computing systems
- ....

Checkpointing

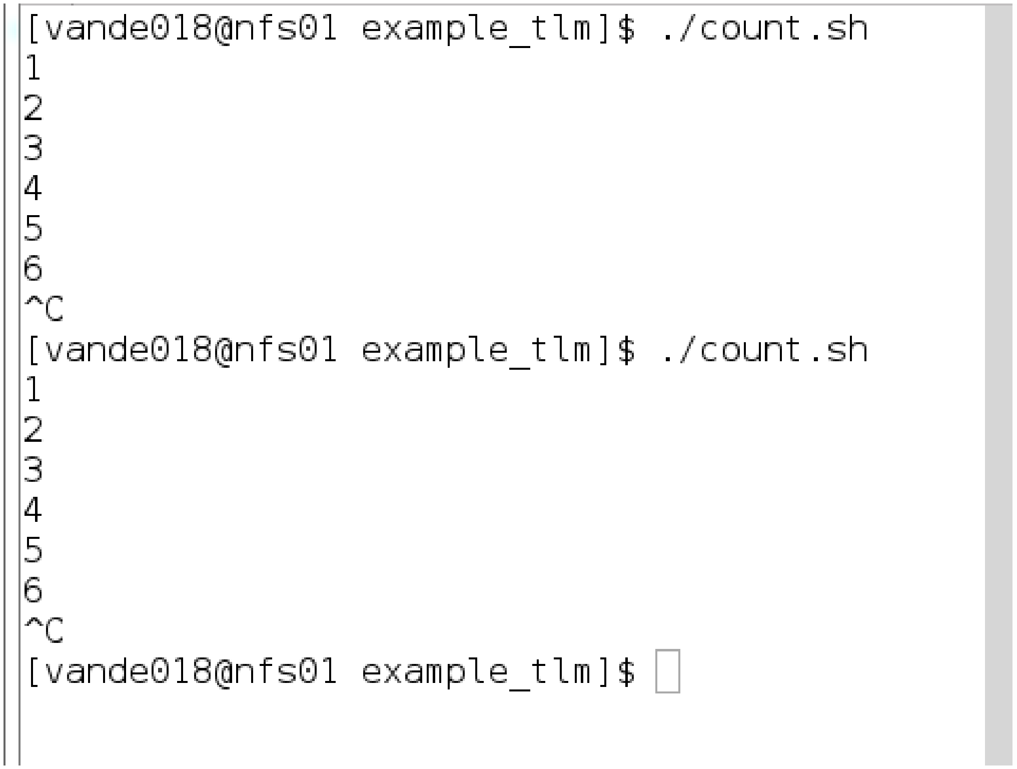
Without checkpointing



```
[vande018@nfs01 example_tlm]$ ./count.sh
1
2
3
4
5
6
^C
[vande018@nfs01 example_tlm]$ ./count.sh
1
2
3
4
5
6
^C
[vande018@nfs01 example_tlm]$ 
```

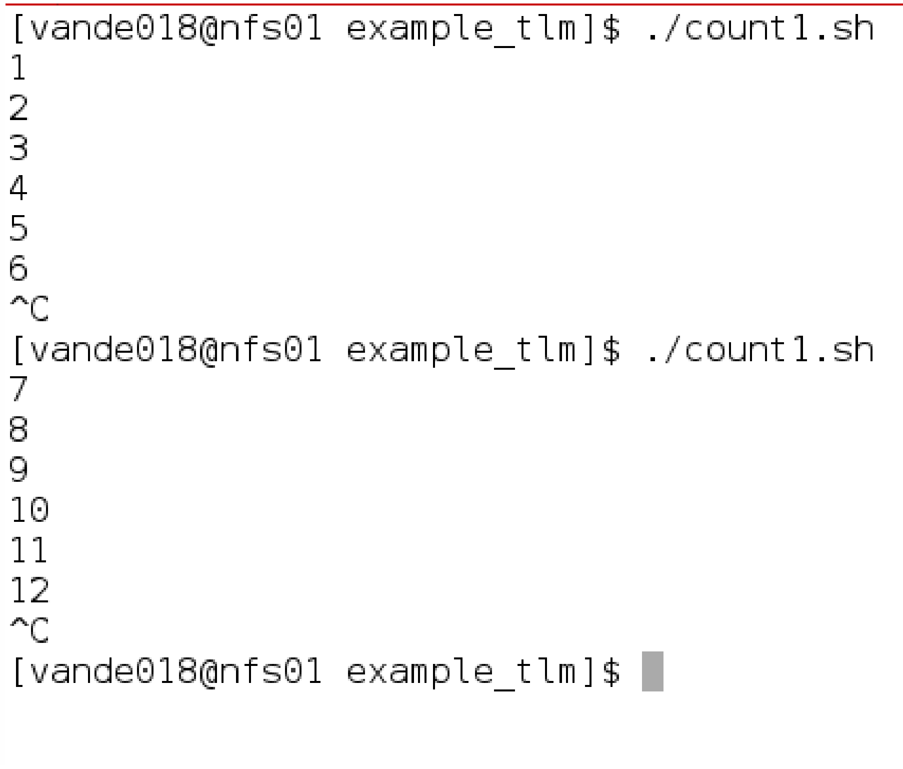
Checkpointing

Without checkpointing



```
[vande018@nfs01 example_tlm]$ ./count.sh
1
2
3
4
5
6
^C
[vande018@nfs01 example_tlm]$ ./count.sh
1
2
3
4
5
6
^C
[vande018@nfs01 example_tlm]$ 
```

With checkpointing



```
[vande018@nfs01 example_tlm]$ ./count1.sh
1
2
3
4
5
6
^C
[vande018@nfs01 example_tlm]$ ./count1.sh
7
8
9
10
11
12
^C
[vande018@nfs01 example_tlm]$ 
```

Two types of program

➔You **have access** to the code

➔You **don't have access** to the code

You **have access** to the code!

You have access to the code

Recipe

Modify the code to implement the following recipe:

1. Look for a state file
    - Includes all information required to restore the state when the program was stopped

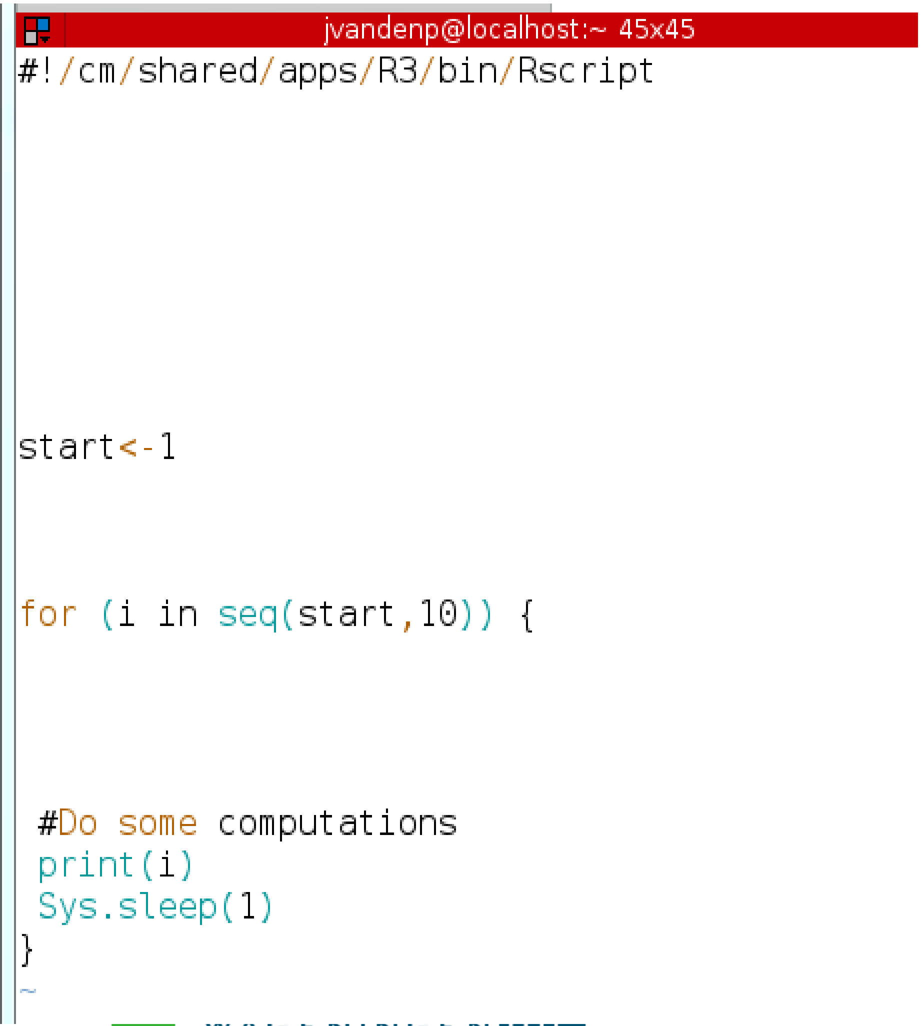
2. If it exists, read it and restore the state
   Else create an initial state

3. Periodically save the state

WAGENINGEN UR
For quality of life

R example

Without checkpointing



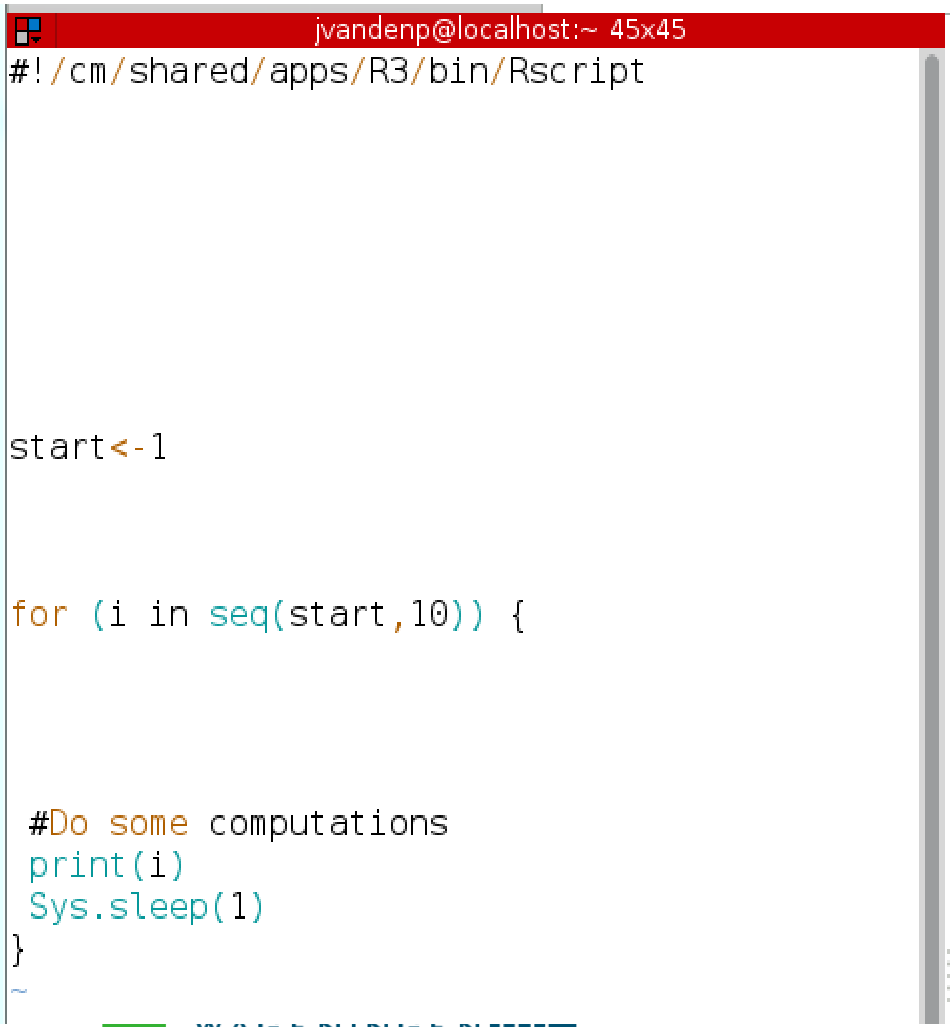
jvandenp@localhost:~ 45x45

```
#!/cm/shared/apps/R3/bin/Rscript
start<-1
for (i in seq(start,10)) {
 #Do some computations
 print(i)
 Sys.sleep(1)
}
~
```

WAGENINGEN UR
For quality of life

R example

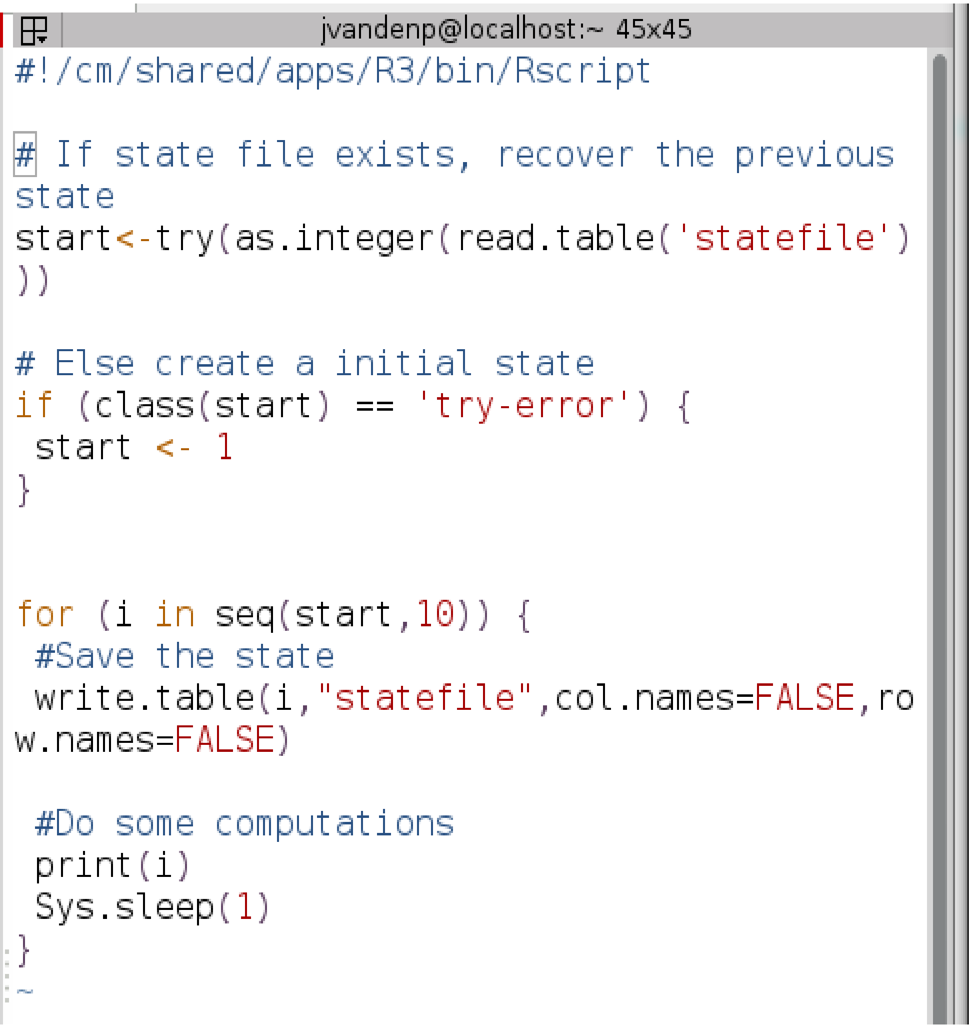
Without checkpointing



jvandenp@localhost:~ 45x45

```
#!/cm/shared/apps/R3/bin/Rscript








start<-1



for (i in seq(start,10)) {




 #Do some computations
 print(i)
 Sys.sleep(1)
}
~
```

With checkpointing



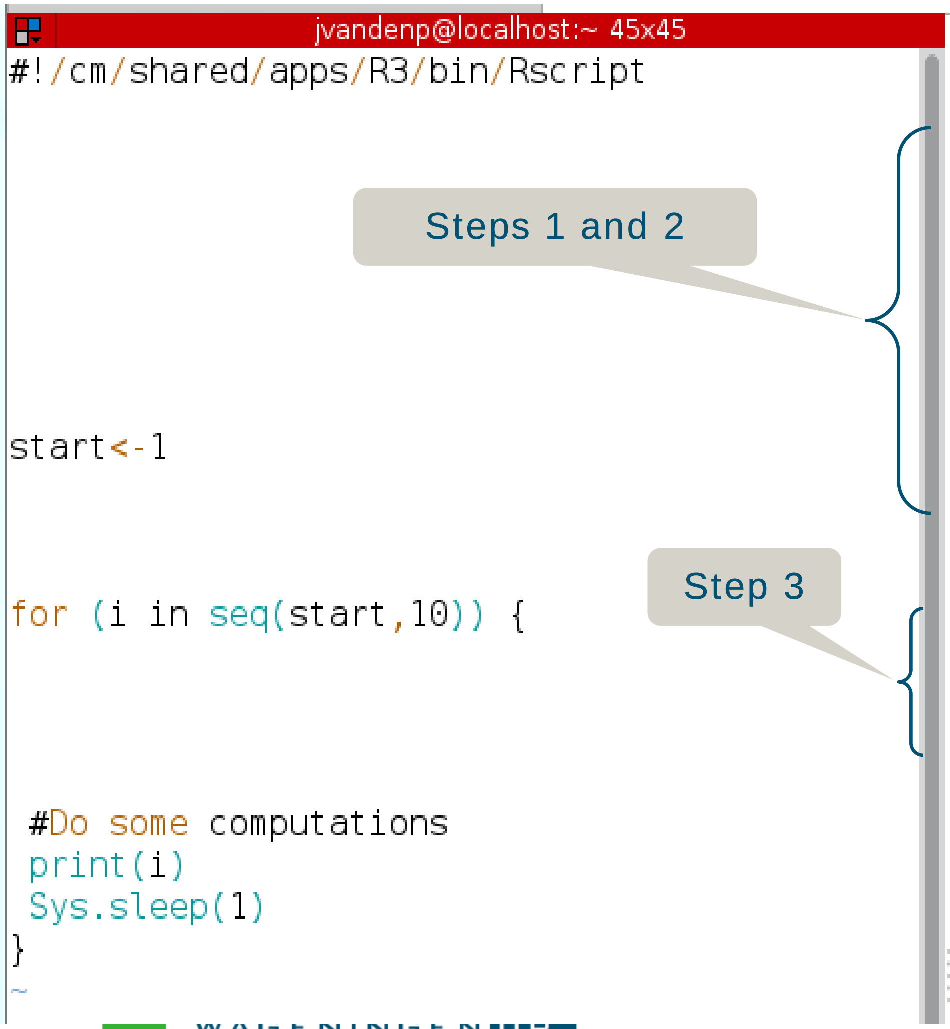
jvandenp@localhost:~ 45x45

```
#!/cm/shared/apps/R3/bin/Rscript

# If state file exists, recover the previous
state
start<-try(as.integer(read.table('statefile')
))

# Else create a initial state
if (class(start) == 'try-error') {
 start <- 1
}


for (i in seq(start,10)) {
 #Save the state
 write.table(i,"statefile",col.names=FALSE,ro
w.names=FALSE)

 #Do some computations
 print(i)
 Sys.sleep(1)
}
~
```

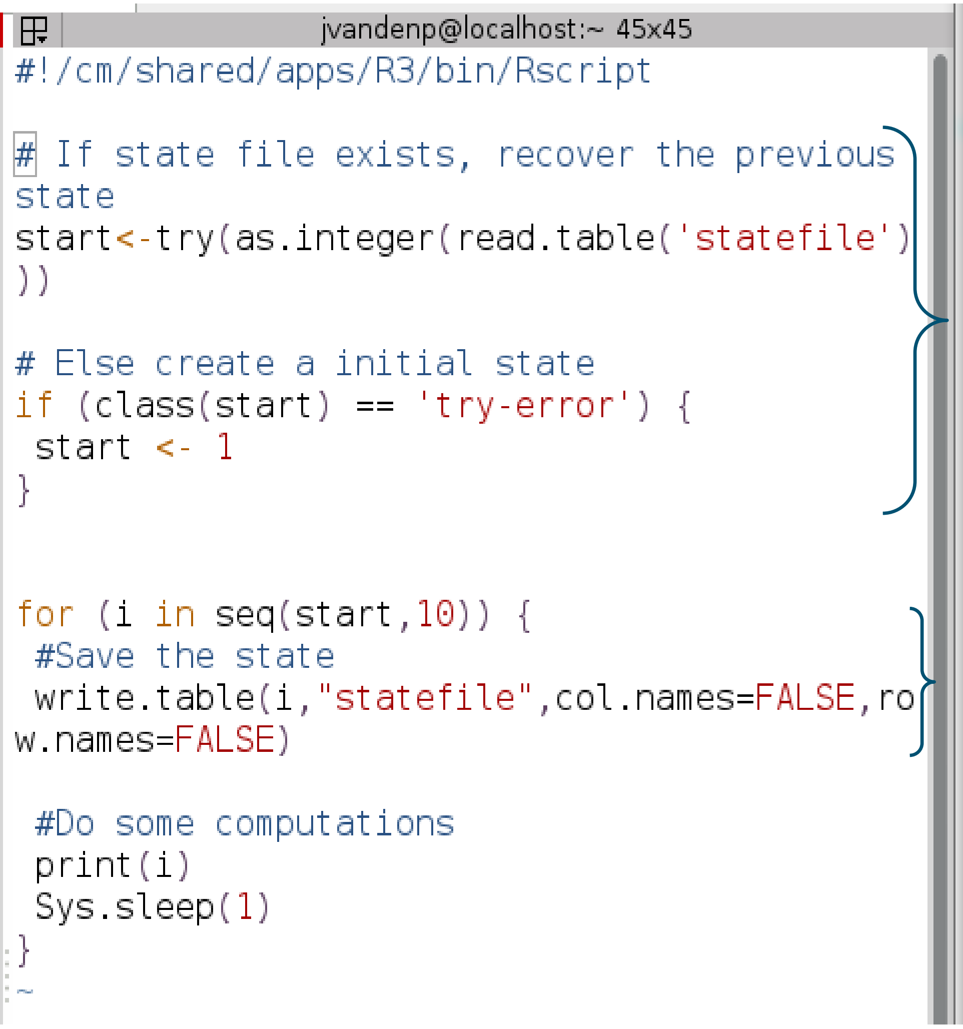
R example

Without checkpointing

With checkpointing



```
#!/cm/shared/apps/R3/bin/Rscript

start<-1

for (i in seq(start,10)) {

 #Do some computations
 print(i)
 Sys.sleep(1)
}
~
```



```
#!/cm/shared/apps/R3/bin/Rscript

# If state file exists, recover the previous
state
start<-try(as.integer(read.table('statefile')
))

# Else create a initial state
if (class(start) == 'try-error') {
 start <- 1
}

for (i in seq(start,10)) {
 #Save the state
 write.table(i,"statefile",col.names=FALSE,ro
w.names=FALSE)

 #Do some computations
 print(i)
 Sys.sleep(1)
}
~
```

WAGENINGEN UR
For quality of life

You have access to the code

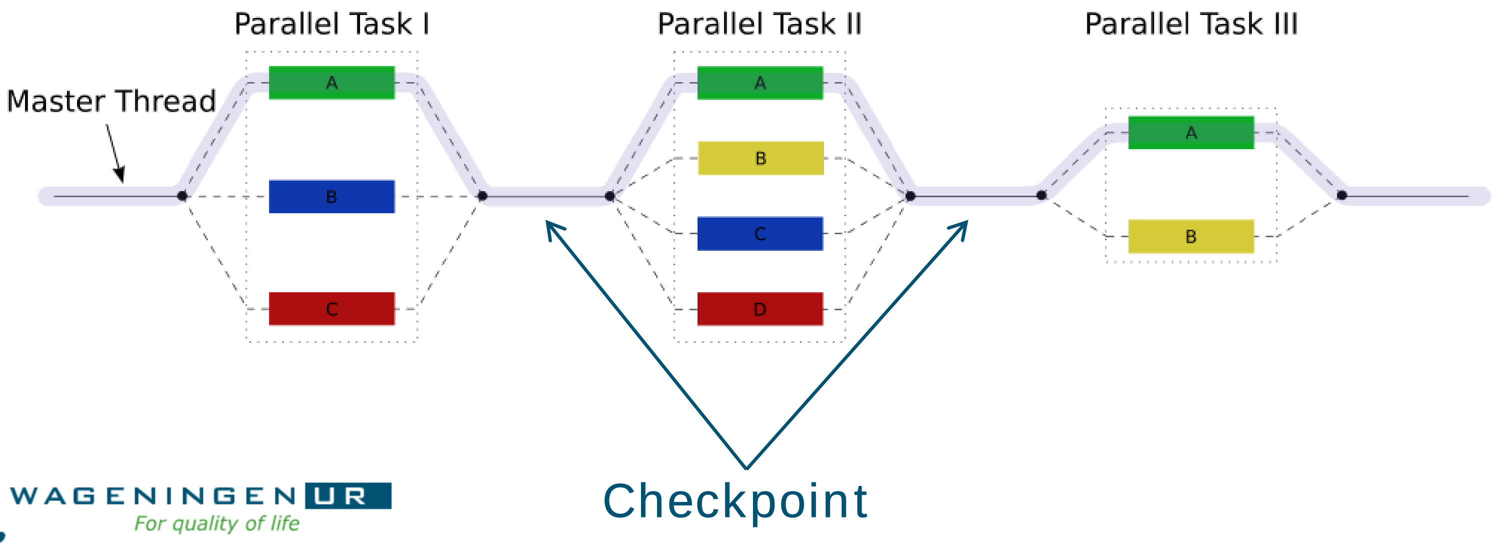
Recipe

- Recipe applicable to several languages
  - R, Python, Matlab (Octave), Fortran, C, shell, ...

You have access to the code

Recipe

- Recipe applicable to several languages
  - R, Python, Matlab (Octave), Fortran, C, shell, ...
- Checkpointing of parallel programs is easier after a global synchronization



You have access to the code
Recipe

- Checkpointing: requirements
  - (Some) efforts (writing additional code)
    - Function of the program
  - Memory (e.g. for the state file)

    ➔Be careful to what is saved
  - Time (e.g., to write the state file)

    ➔Avoid to checkpoint too often

    ➔Use Slurm and other software features

You have access to the code

Slurm and other software features

- Software features
  - E.g., R --restore <script.R
- Slurm features
  - Slurm can send signals
    - scancel --signal USR1 $JOB_ID
    - sbatch --signal=INT@120

  ➔Modify the program to look periodically to this signal

  ➔If the signal is received, checkpoint and exit

You **don’t have access** to the code!

You don’t have access to the code

- Many software are checkpointable

➔ **Read** the manual!

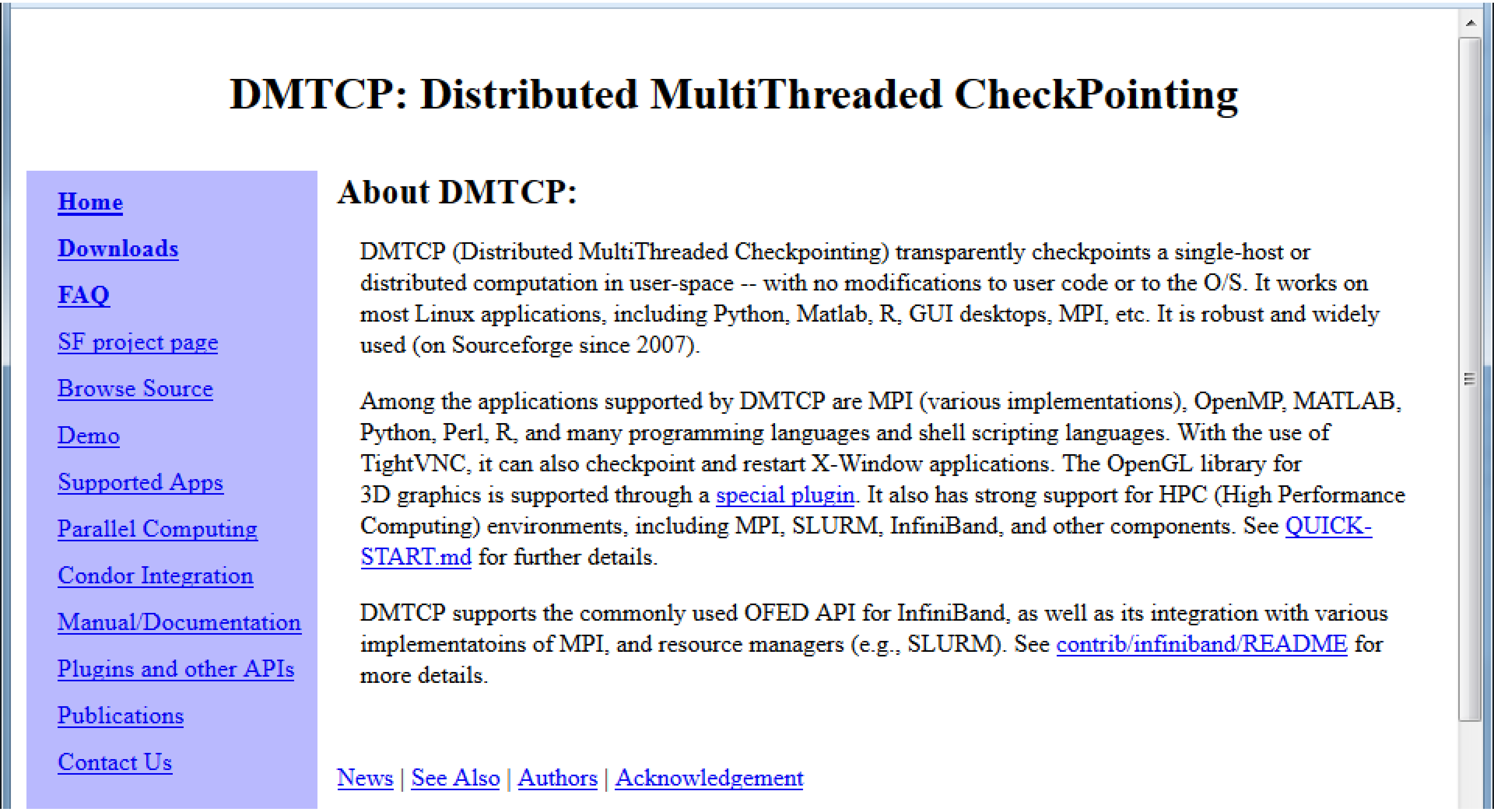
➔ **If it is checkpointable**, adapt the (Slurm) script!

➔ **If it is not checkpointable**, some software could help you!

  - E.g., DMTCP

WAGENINGEN UR
For quality of life

DMTCP



DMTCP: Distributed MultiThreaded CheckPointing

Home
Downloads
FAQ
SF project page
Browse Source
Demo
Supported Apps
Parallel Computing
Condor Integration
Manual/Documentation
Plugins and other APIs
Publications
Contact Us

About DMTCP:

DMTCP (Distributed MultiThreaded Checkpointing) transparently checkpoints a single-host or distributed computation in user-space -- with no modifications to user code or to the O/S. It works on most Linux applications, including Python, Matlab, R, GUI desktops, MPI, etc. It is robust and widely used (on Sourceforge since 2007).

Among the applications supported by DMTCP are MPI (various implementations), OpenMP, MATLAB, Python, Perl, R, and many programming languages and shell scripting languages. With the use of TightVNC, it can also checkpoint and restart X-Window applications. The OpenGL library for 3D graphics is supported through a special plugin. It also has strong support for HPC (High Performance Computing) environments, including MPI, SLURM, InfiniBand, and other components. See QUICK-START.md for further details.

DMTCP supports the commonly used OFED API for InfiniBand, as well as its integration with various implementatoins of MPI, and resource managers (e.g., SLURM). See contrib/infiniband/README for more details.

News | See Also | Authors | Acknowledgement

http://dmtcp.sourceforge.net/

DMTCP

- Linux applications
  - R, Matlab, Java, Python, Fortran, ...
- Sequential and parallel computations
- No modification to the code or OS (no root privilege)
- Requires
  - Independent monitoring process
  - A shared library

DMTCP
Recipe

1. Load the module DMTCP
2. Run a non-checkpointable program with dmtcp_launch
   - Required commands:
     - dmtcp_launch
     - dmtcp_command

   ➔At each checkpoint, DMTCP creates required statefiles
3. Restart it with dmtcp_restart_script.sh

➔ Must be used with SLURM on the HPC!

WAGENINGEN UR
For quality of life

DMTCP
Recipe for SLURM

- Two SLURM batch scripts
  - Run the first time the program (step 2)
  - Restart the program if it was checkpointed (step 3)

OR

- One SLURM batch script to do both tasks (steps 2 and 3)

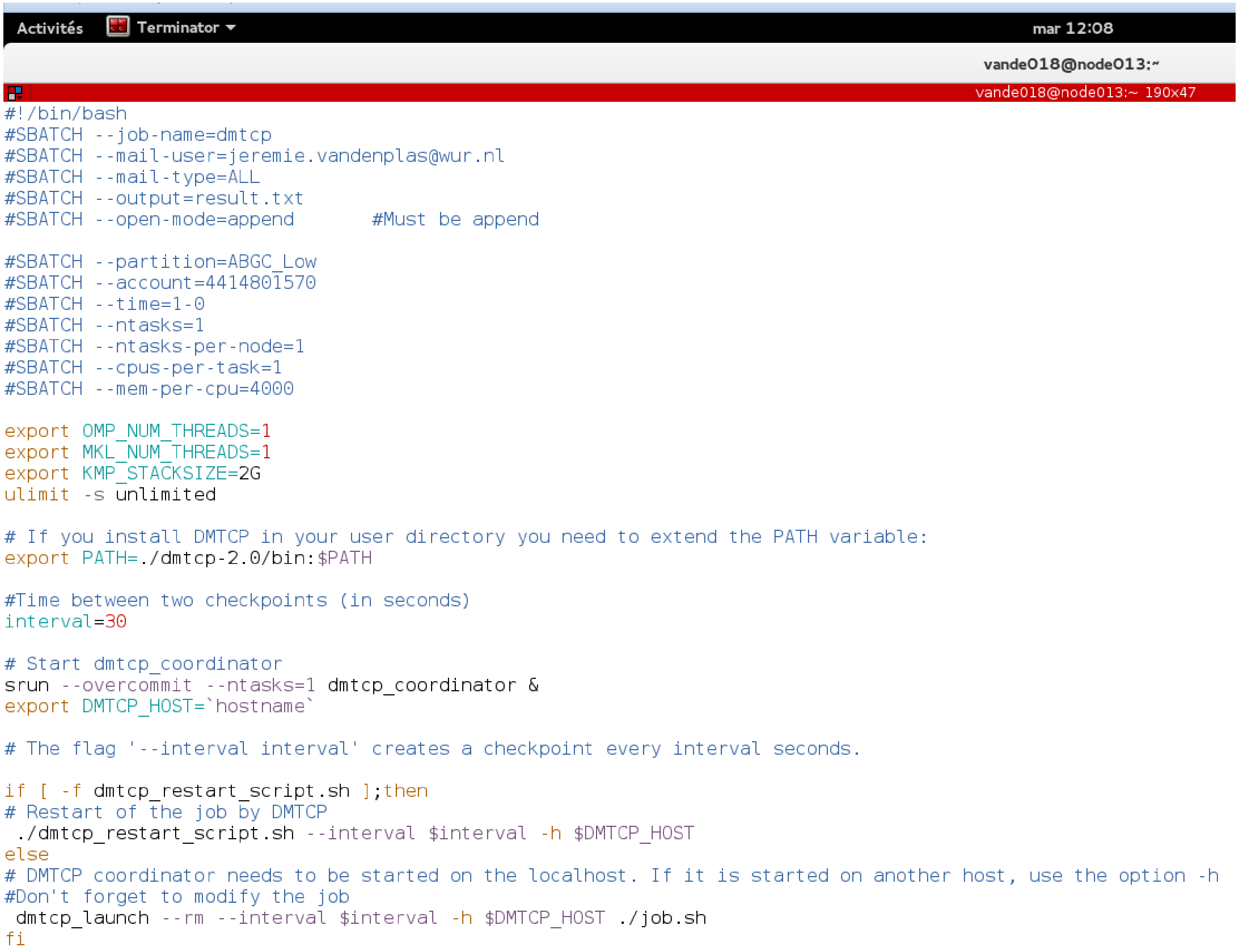
DMTCP
SLURM batch script

- Example of SLURM script

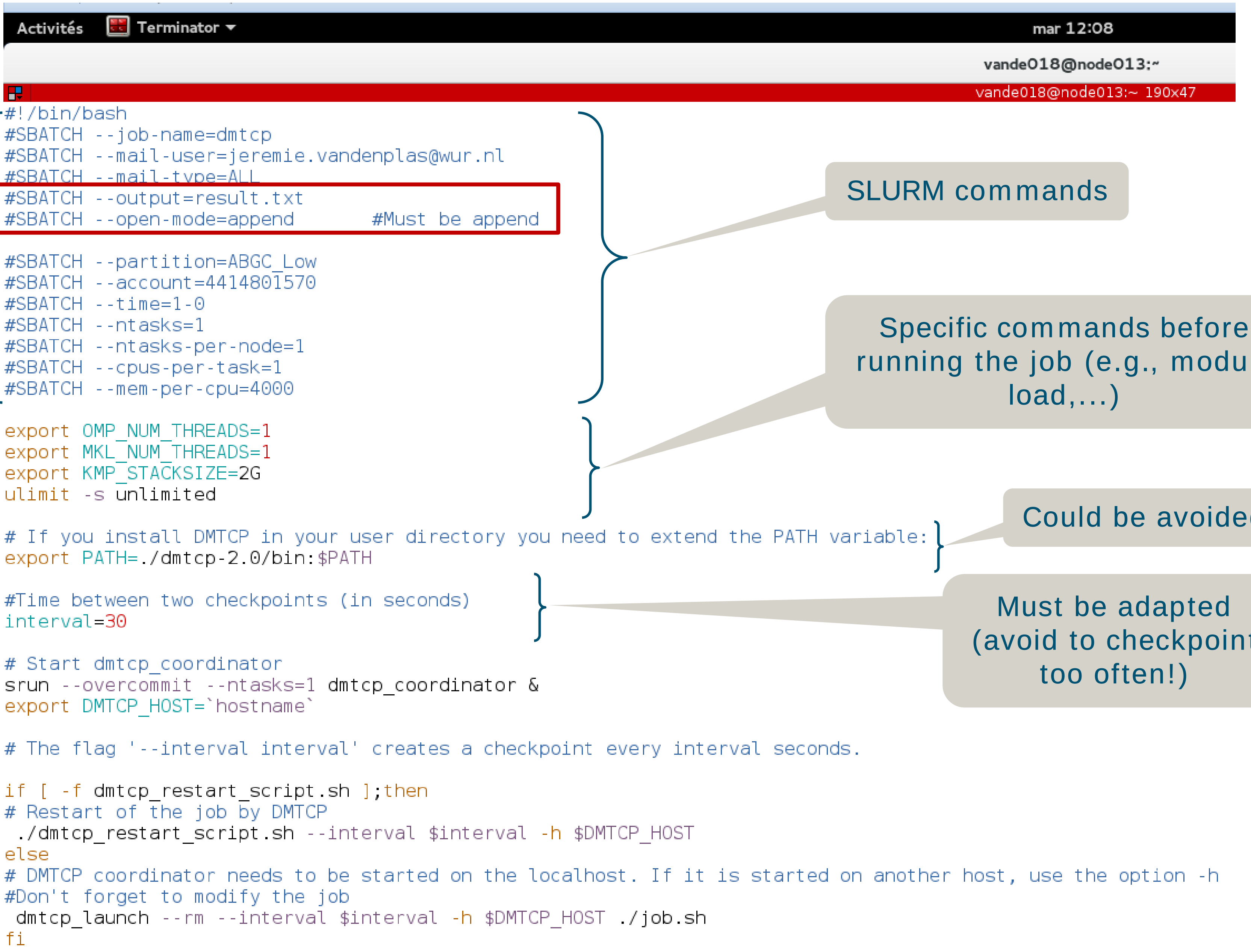
  /cm/shared/apps/dmtcp/gcc/64/current/examples

- An example

WAGENINGEN UR
For quality of life



```
#!/bin/bash
#SBATCH --job-name=dmtcp
#SBATCH --mail-user=jeremie.vandenplas@wur.nl
#SBATCH --mail-type=ALL
#SBATCH --output=result.txt
#SBATCH --open-mode=append       #Must be append

#SBATCH --partition=ABGC_Low
#SBATCH --account=4414801570
#SBATCH --time=1-0
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export KMP_STACKSIZE=2G
ulimit -s unlimited

# If you install DMTCP in your user directory you need to extend the PATH variable:
export PATH=./dmtcp-2.0/bin:$PATH

#Time between two checkpoints (in seconds)
interval=30

# Start dmtcp_coordinator
srun --overcommit --ntasks=1 dmtcp_coordinator &
export DMTCP_HOST=`hostname`

# The flag '--interval interval' creates a checkpoint every interval seconds.

if [ -f dmtcp_restart_script.sh ];then
# Restart of the job by DMTCP
 ./dmtcp_restart_script.sh --interval $interval -h $DMTCP_HOST
else
# DMTCP coordinator needs to be started on the localhost. If it is started on another host, use the option -h
#Don't forget to modify the job
 dmtcp_launch --rm --interval $interval -h $DMTCP_HOST ./job.sh
fi
```



```
#!/bin/bash
#SBATCH --job-name=dmtcp
#SBATCH --mail-user=jeremie.vandenplas@wur.nl
#SBATCH --mail-type=ALL
#SBATCH --output=result.txt
#SBATCH --open-mode=append        #Must be append

#SBATCH --partition=ABGC_Low
#SBATCH --account=4414801570
#SBATCH --time=1-0
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export KMP_STACKSIZE=2G
ulimit -s unlimited

# If you install DMTCP in your user directory you need to extend the PATH variable:
export PATH=./dmtcp-2.0/bin:$PATH

#Time between two checkpoints (in seconds)
interval=30

# Start dmtcp_coordinator
srun --overcommit --ntasks=1 dmtcp_coordinator &
export DMTCP_HOST=`hostname`

# The flag '--interval interval' creates a checkpoint every interval seconds.

if [ -f dmtcp_restart_script.sh ];then
# Restart of the job by DMTCP
 ./dmtcp_restart_script.sh --interval $interval -h $DMTCP_HOST
else
# DMTCP coordinator needs to be started on the localhost. If it is started on another host, use the option -h
#Don't forget to modify the job
 dmtcp_launch --rm --interval $interval -h $DMTCP_HOST ./job.sh
fi
```

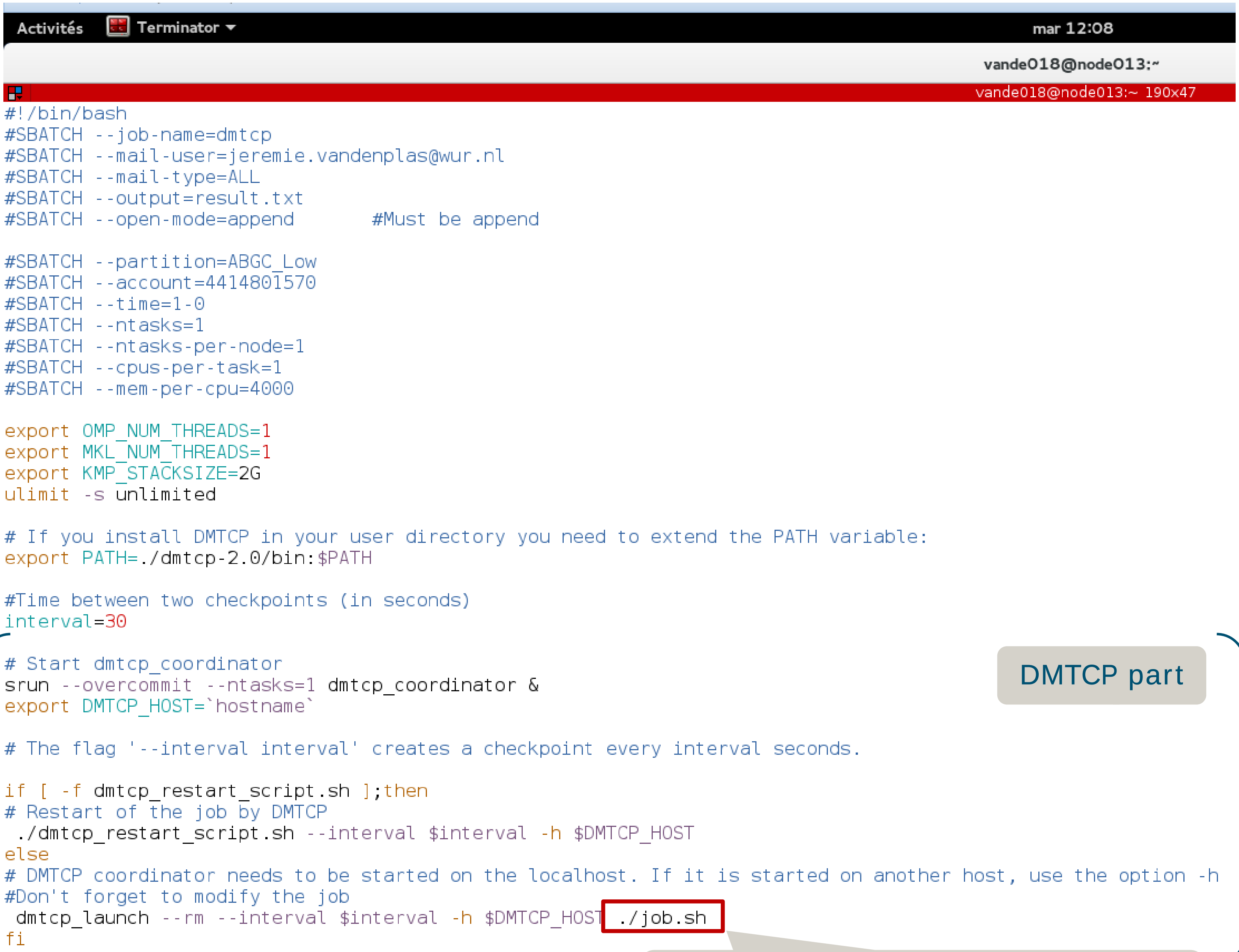
SLURM commands

Specific commands before running the job (e.g., module load,...)

Could be avoided

Must be adapted (avoid to checkpoint too often!)

WAGENINGEN UR
For quality of life



```
#!/bin/bash
#SBATCH --job-name=dmtcp
#SBATCH --mail-user=jeremie.vandenplas@wur.nl
#SBATCH --mail-type=ALL
#SBATCH --output=result.txt
#SBATCH --open-mode=append       #Must be append

#SBATCH --partition=ABGC_Low
#SBATCH --account=4414801570
#SBATCH --time=1-0
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=4000

export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export KMP_STACKSIZE=2G
ulimit -s unlimited

# If you install DMTCP in your user directory you need to extend the PATH variable:
export PATH=./dmtcp-2.0/bin:$PATH

#Time between two checkpoints (in seconds)
interval=30

# Start dmtcp_coordinator
srun --overcommit --ntasks=1 dmtcp_coordinator &
export DMTCP_HOST=`hostname`

# The flag '--interval interval' creates a checkpoint every interval seconds.

if [ -f dmtcp_restart_script.sh ];then
# Restart of the job by DMTCP
 ./dmtcp_restart_script.sh --interval $interval -h $DMTCP_HOST
else
# DMTCP coordinator needs to be started on the localhost. If it is started on another host, use the option -h
#Don't forget to modify the job
 dmtcp_launch --rm --interval $interval -h $DMTCP_HOST ./job.sh
fi
```

DMTCP part

Only this part must be adapted with command for the actual job!

WAGENINGEN UR
For quality of life

Summary

- If you **have access** to the code…

  … Make it checkpointable

- If you **don’t have access** to the code…
  - Verify if it is checkpointable
  - If it is not checkpointable

    ➔Solutions may exist (e.g., DMTCP)

Thank you!

Questions?