UCLouvain

Plateforme technologique de Calcul Intensif et Stockage de Masse

EURO BELGIUM

Checkpointing

Olivier Mattelaer

What is checkpointing

?

```
$ ./count
$ ./count
1
```

```
$ ./count
1
2
```

```
$ ./count
1
2
3
```

```
$ ./count
1
2
3^C
$
```

```
$ ./count
1
2
3^C
$ ./count
$ ./count
1
2
3^C
$ ./count
1
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
2
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
5
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
2
3
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
5
6
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
2
3
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
5
6
```

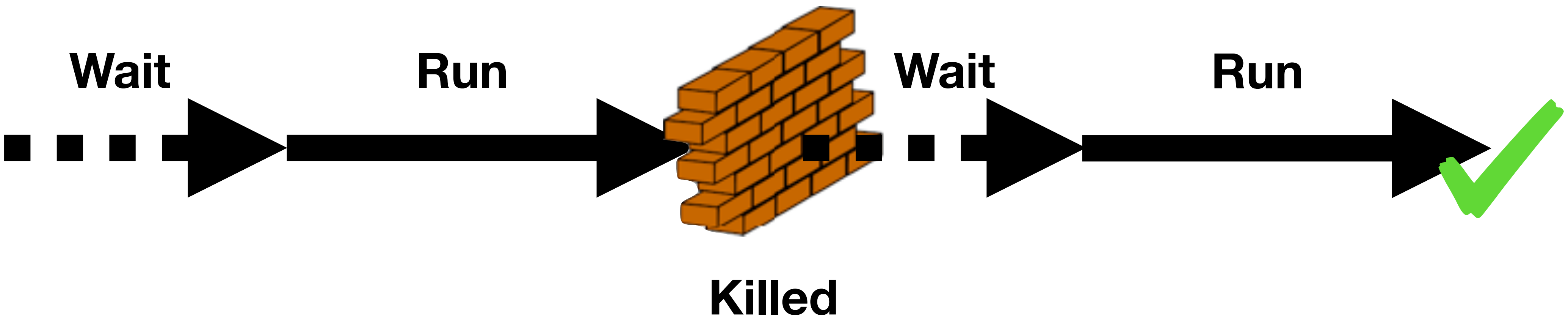
Checkpointing:

'saving' a computation
so that it can be resumed later
(rather than started again)

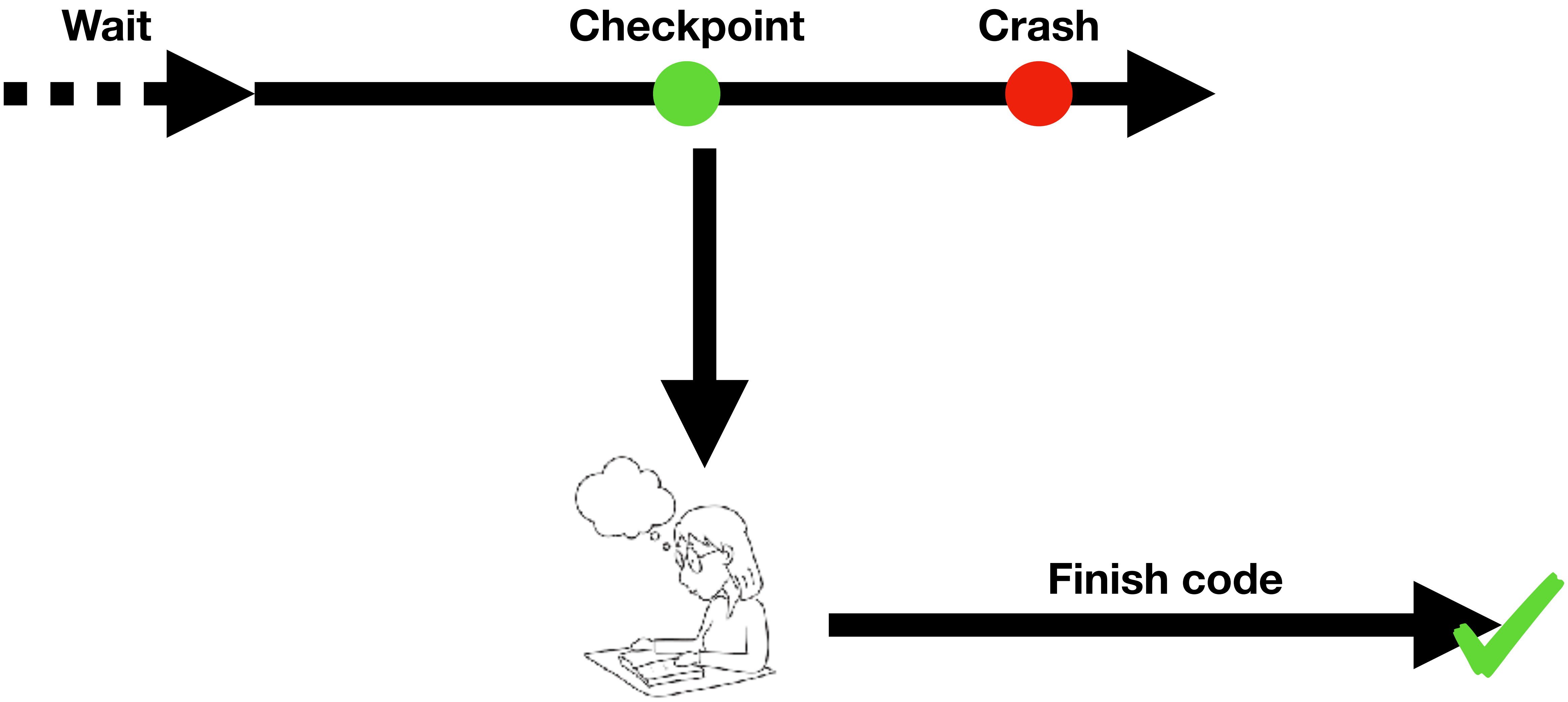
Why do we need checkpointing

?

Wall-Time



Debug



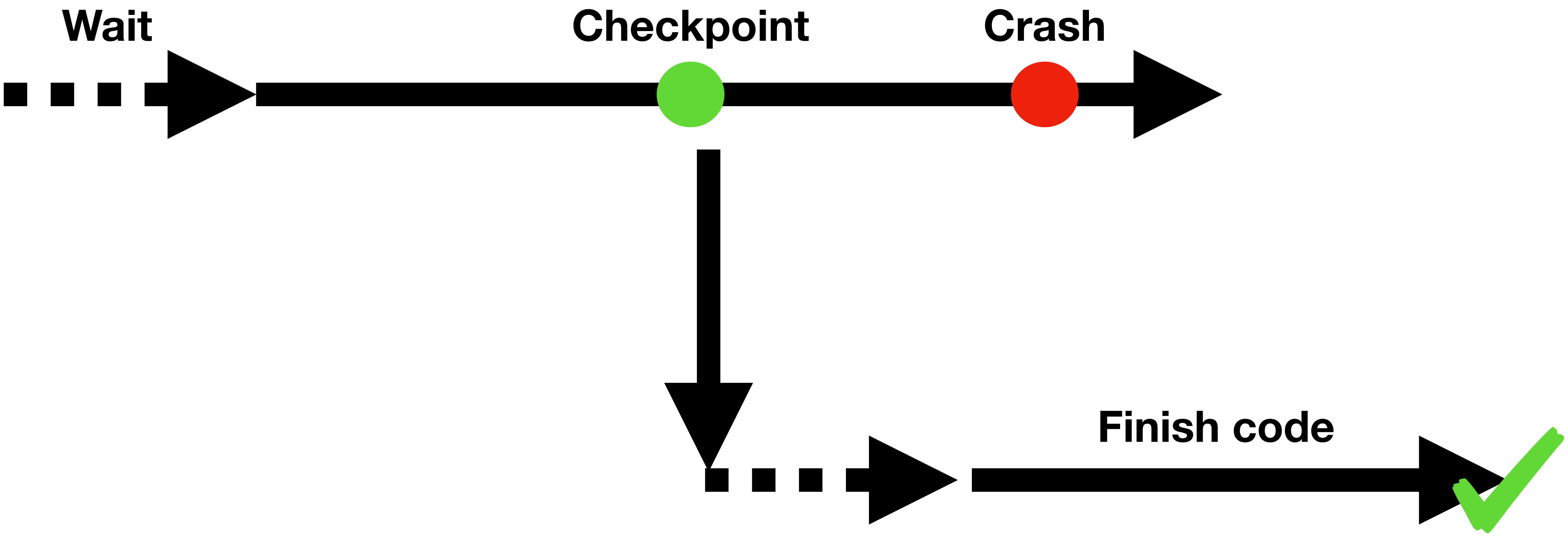
Wait

Checkpoint

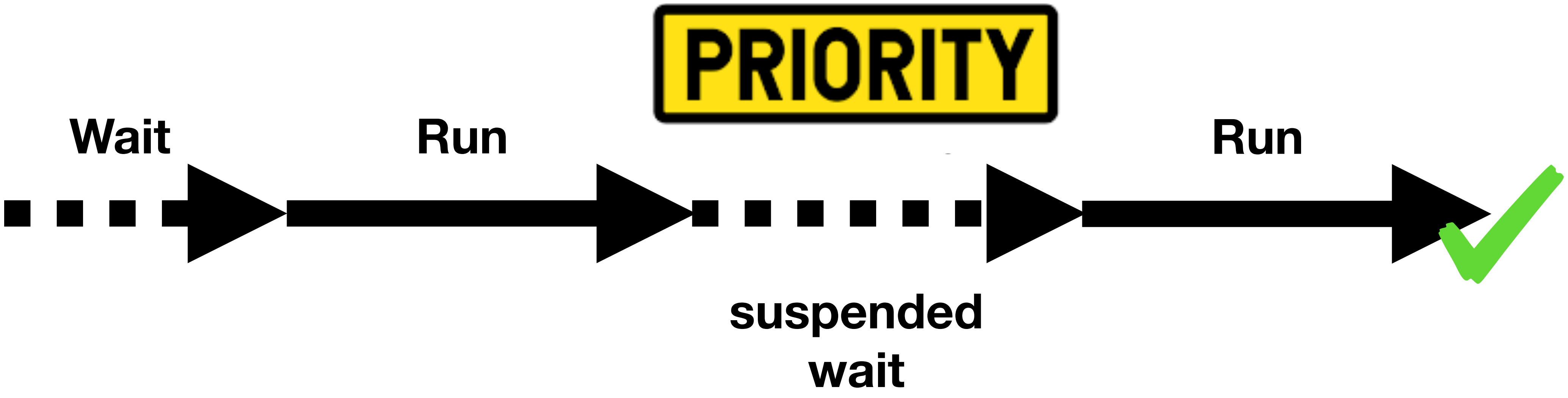
Crash

Finish code

Hardware crash



Pre-emption



Today Agenda

- How to checkpoint every **iteration**.
  - Easy just setting the stage
- How to checkpoint **on demand**.
  - Signal
- How to checkpoint **every X minutes**
  - When you can not change the program

Working with checkpoint-restart-able software

Many scientific software have built-in checkpointing capabilities

(although it might not be called that way)

Check the documentation

Demo #1

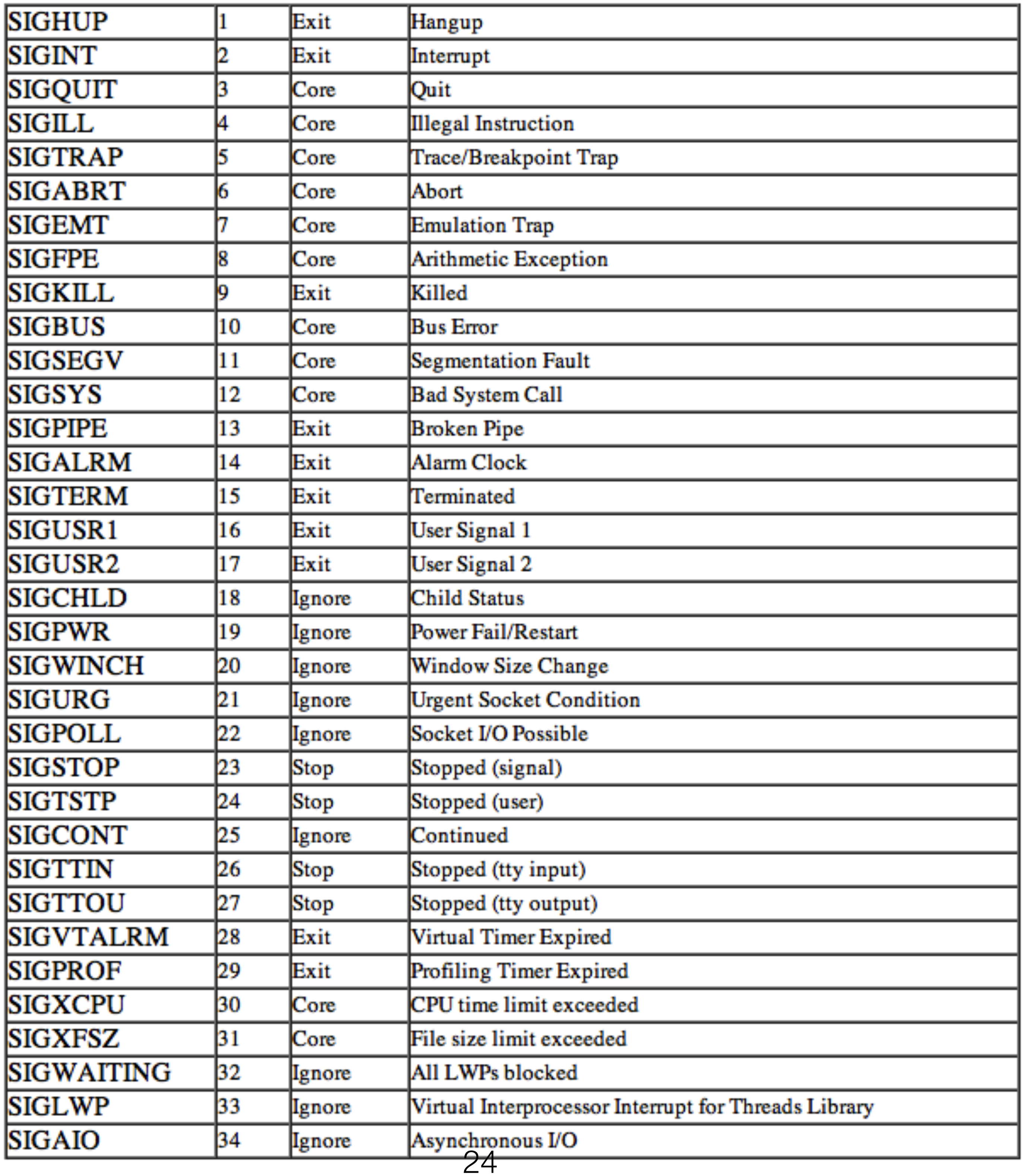
count.py

Save state at each iteration

2

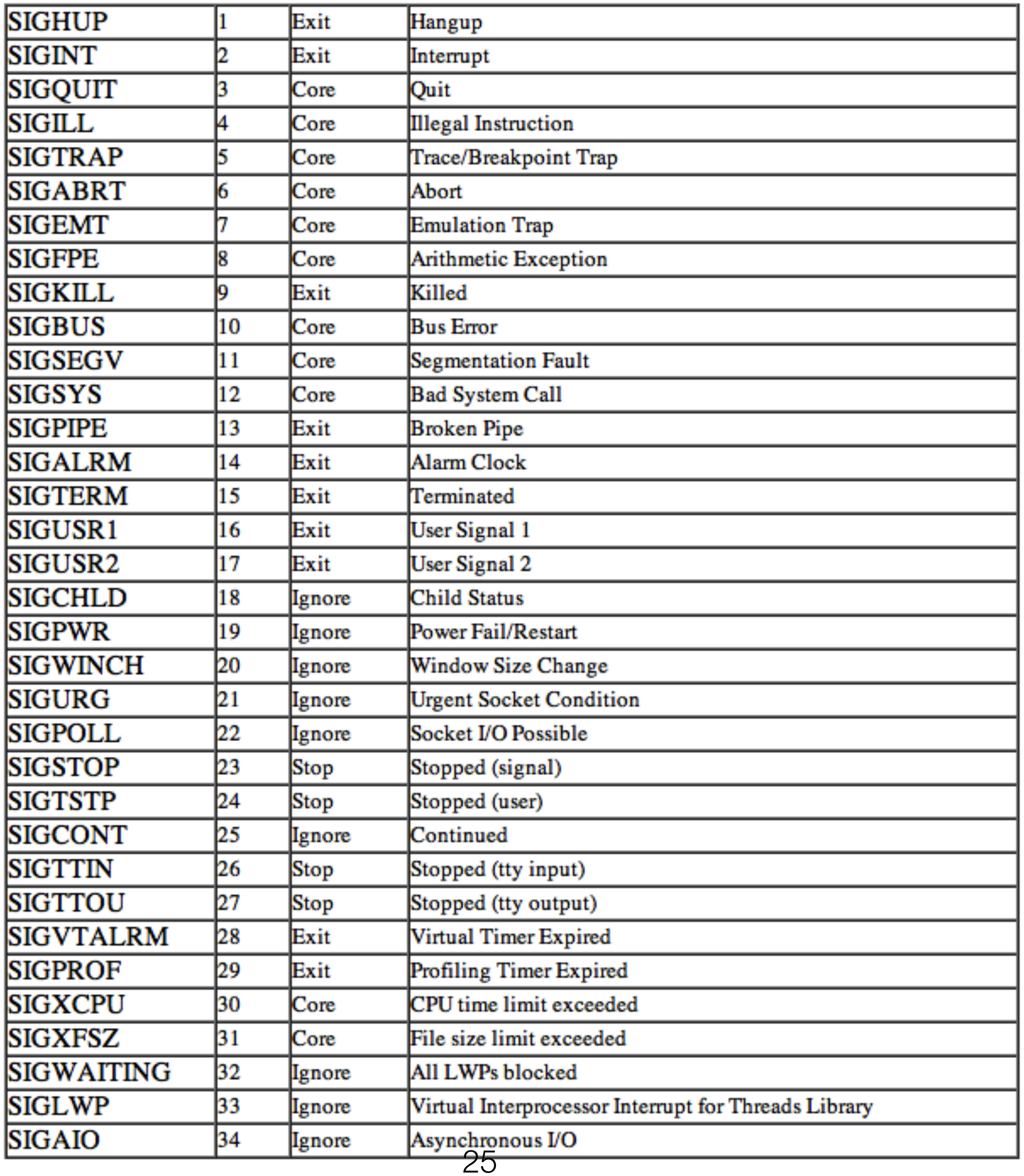
Using UNIX signals to reduce overhead : do not save the state at each iteration -- wait for the signal.

UNIX processes can receive 'signals' from the user, the OS, or another process



SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

UNIX processes can receive 'signals' from the user, the OS, or another process



SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

^C

^D

kill -9

kill

^Z

fg, bg

Demo #2

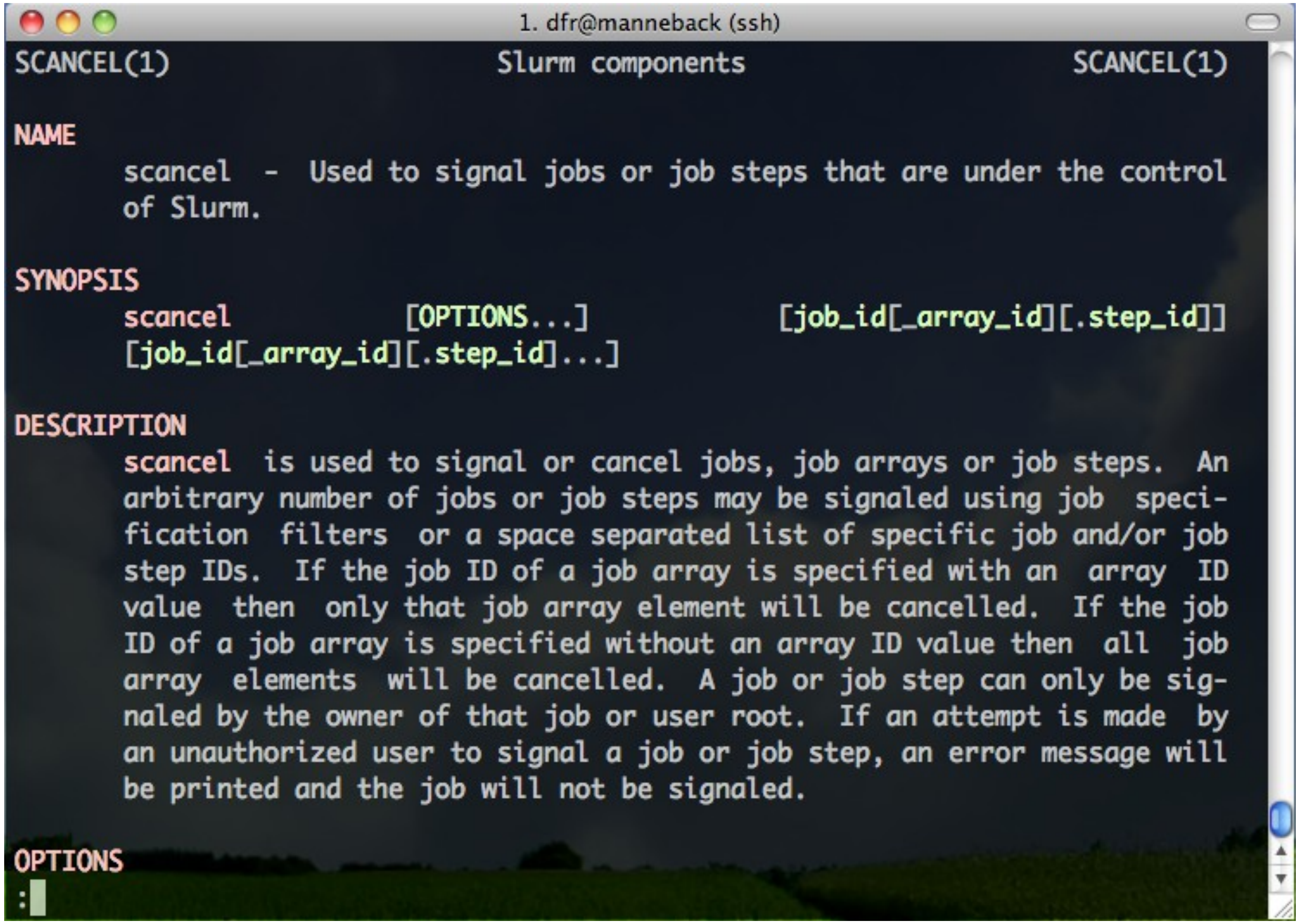
count-signal.py

Catch control-C to save state

3

Use Slurm signaling abilities to manage checkpoint-able software in Slurm scripts on the clusters.

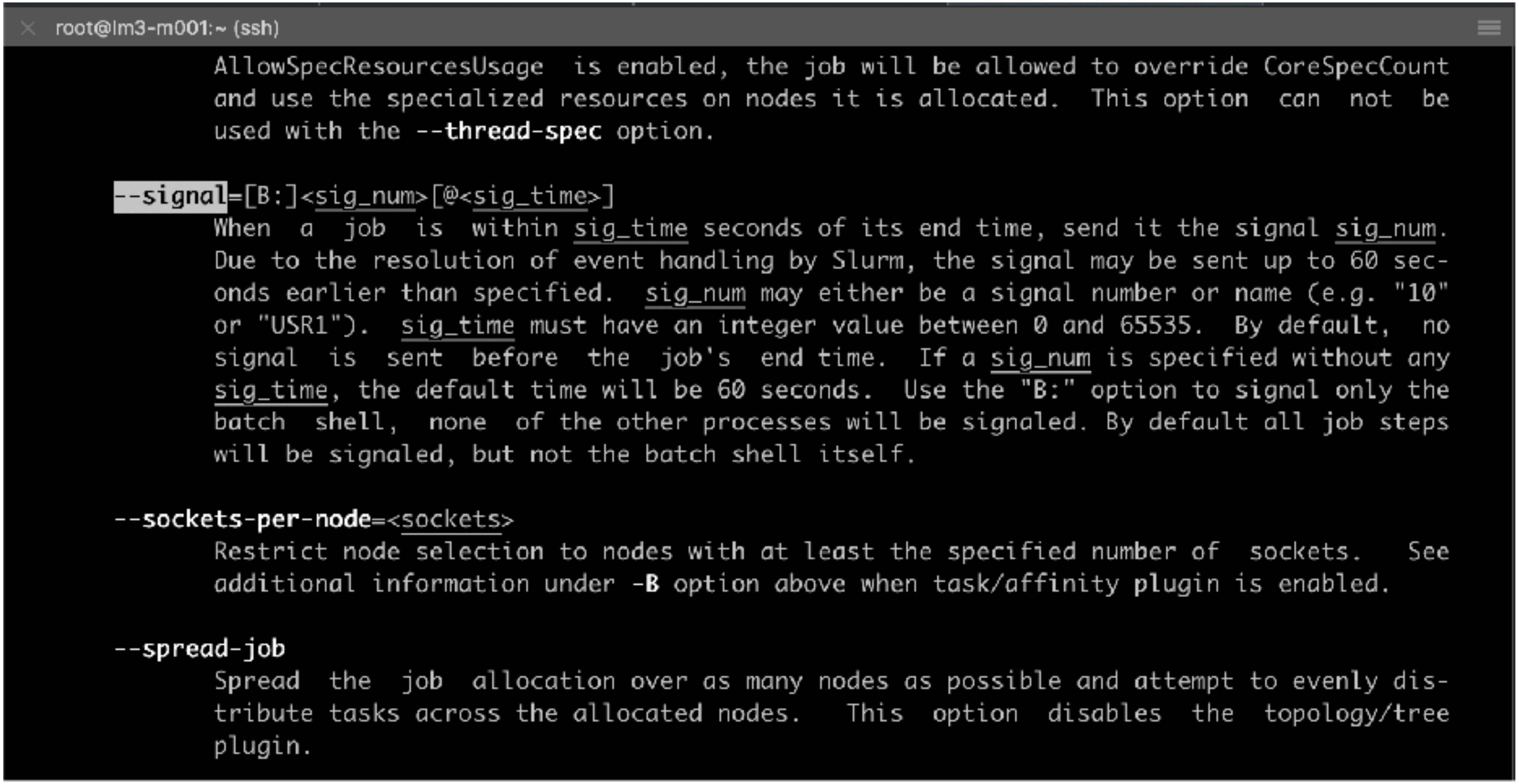
scancel is used to send signals to jobs



```
SCANCEL(1)                      Slurm components                      SCANCEL(1)

NAME
       scancel  -  Used to signal jobs or job steps that are under the control
       of Slurm.

SYNOPSIS
       scancel            [OPTIONS...]            [job_id[_array_id][.step_id]]
       [job_id[_array_id][.step_id]...]

DESCRIPTION
       scancel  is used to signal or cancel jobs, job arrays or job steps.  An
       arbitrary number of jobs or job steps may be signaled using job  speci-
       fication  filters  or a space separated list of specific job and/or job
       step IDs.  If the job ID of a job array is specified with an  array  ID
       value  then  only that job array element will be cancelled.  If the job
       ID of a job array is specified without an array ID value then  all  job
       array  elements  will be cancelled.  A job or job step can only be sig-
       naled by the owner of that job or user root.  If an attempt is made  by
       an unauthorized user to signal a job or job step, an error message will
       be printed and the job will not be signaled.

OPTIONS
:
```

scancel -s SIGINT JOBID

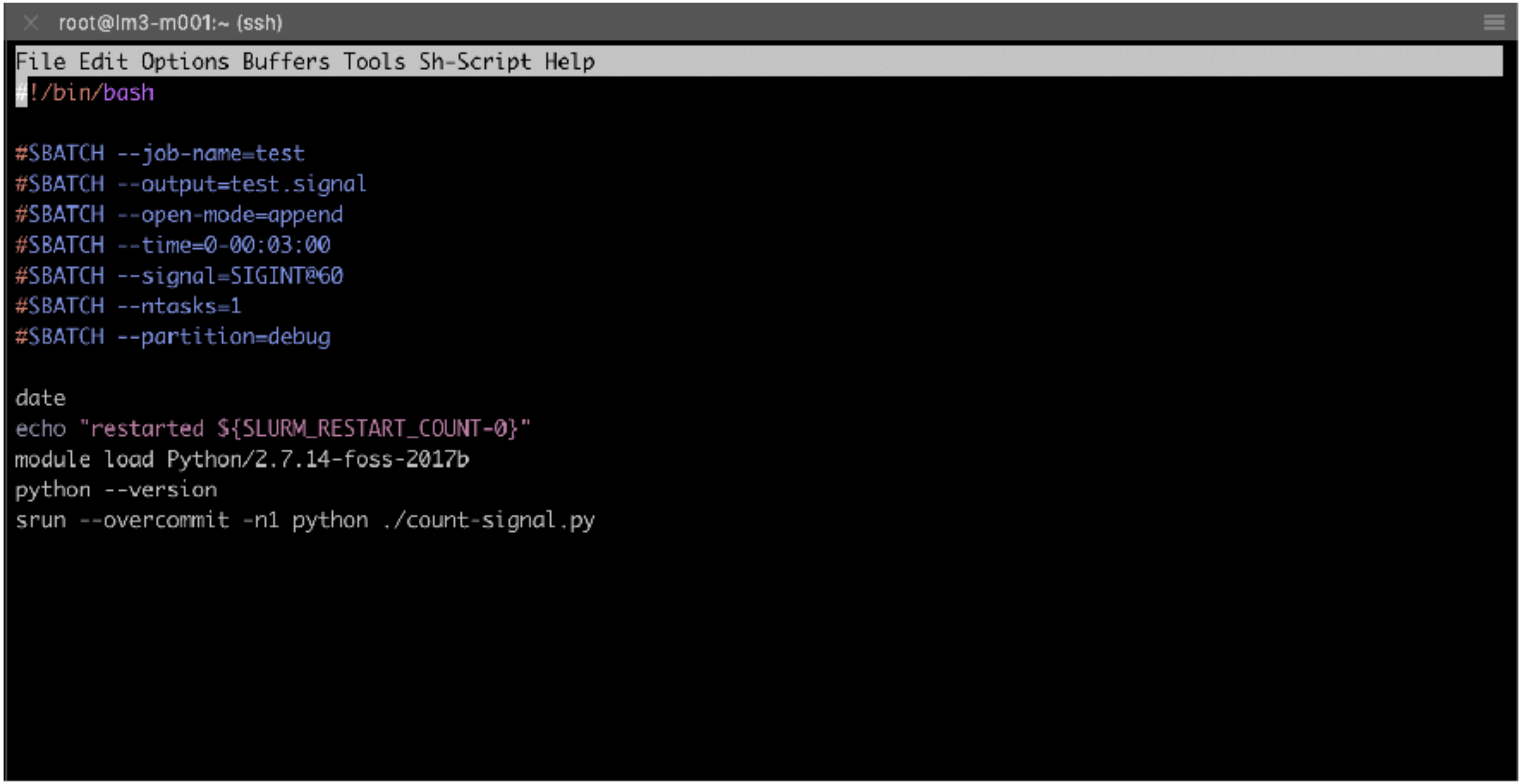
--signal to have Slurm send signals automatically before the end of the allocation



```
              AllowSpecResourcesUsage  is enabled, the job will be allowed to override CoreSpecCount
              and use the specialized resources on nodes it is allocated.  This option  can  not  be
              used with the --thread-spec option.

       --signal=[B:]<sig_num>[@<sig_time>]
              When  a  job  is  within sig_time seconds of its end time, send it the signal sig_num.
              Due to the resolution of event handling by Slurm, the signal may be sent up to 60 sec-
              onds earlier than specified.  sig_num may either be a signal number or name (e.g. "10"
              or "USR1").  sig_time must have an integer value between 0 and 65535.  By default,  no
              signal  is  sent  before  the  job's  end time.  If a sig_num is specified without any
              sig_time, the default time will be 60 seconds.  Use the "B:" option to signal only the
              batch  shell,  none  of the other processes will be signaled. By default all job steps
              will be signaled, but not the batch shell itself.

       --sockets-per-node=<sockets>
              Restrict node selection to nodes with at least the specified number of  sockets.   See
              additional information under -B option above when task/affinity plugin is enabled.

       --spread-job
              Spread  the  job  allocation over as many nodes as possible and attempt to evenly dis-
              tribute tasks across the allocated nodes.   This  option  disables  the  topology/tree
              plugin.
```

--signal=B:SIGINT send signal to the bash script
--signal=SIGINT send signal to the srun command

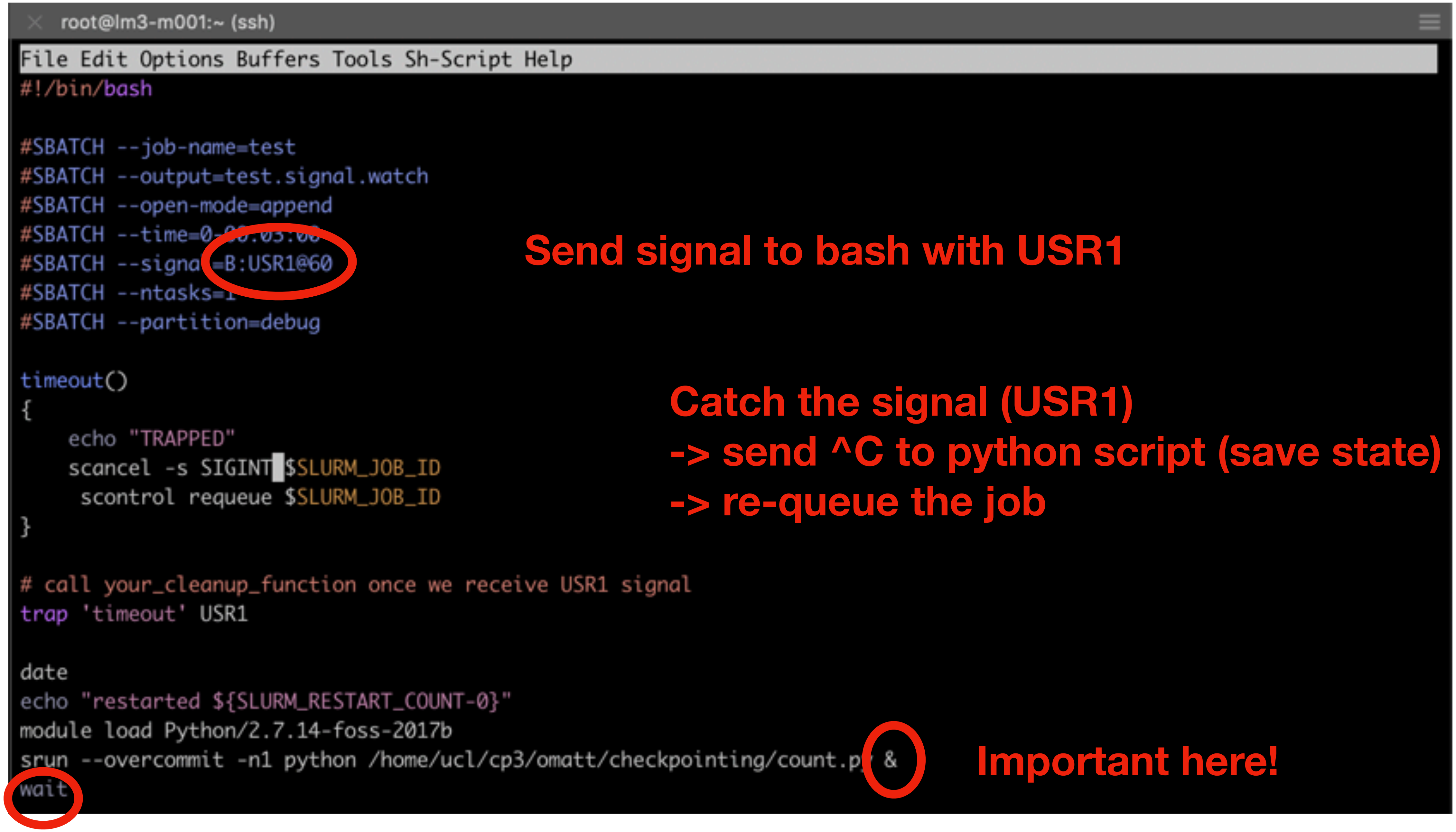
Note the --open-mode=append



```
#!/bin/bash

#SBATCH --job-name=test
#SBATCH --output=test.signal
#SBATCH --open-mode=append
#SBATCH --time=0-00:03:00
#SBATCH --signal=SIGINT@60
#SBATCH --ntasks=1
#SBATCH --partition=debug

date
echo "restarted ${SLURM_RESTART_COUNT-0}"
module load Python/2.7.14-foss-2017b
python --version
srun --overcommit -n1 python ./count-signal.py
```

Note that we need the srun here

Adding requeuing automatically



```
#!/bin/bash

#SBATCH --job-name=test
#SBATCH --output=test.signal.watch
#SBATCH --open-mode=append
#SBATCH --time=0-00:03:00
#SBATCH --signal=B:USR1@60
#SBATCH --ntasks=1
#SBATCH --partition=debug

timeout()
{
    echo "TRAPPED"
    scancel -s SIGINT $SLURM_JOB_ID
     scontrol requeue $SLURM_JOB_ID
}

# call your_cleanup_function once we receive USR1 signal
trap 'timeout' USR1

date
echo "restarted ${SLURM_RESTART_COUNT-0}"
module load Python/2.7.14-foss-2017b
srun --overcommit -n1 python /home/ucl/cp3/omatt/checkpointing/count.py &
wait
```

Send signal to bash with USR1

Catch the signal (USR1)
-> send ^C to python script (save state)
-> re-queue the job

Important here!

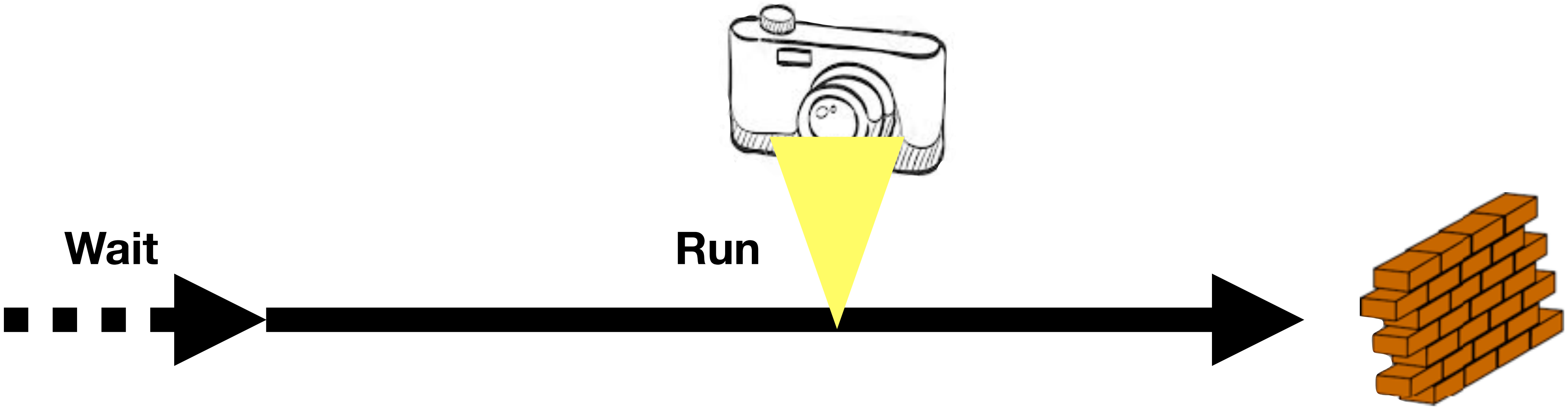
Demo #3

slurm-signal-3.sh

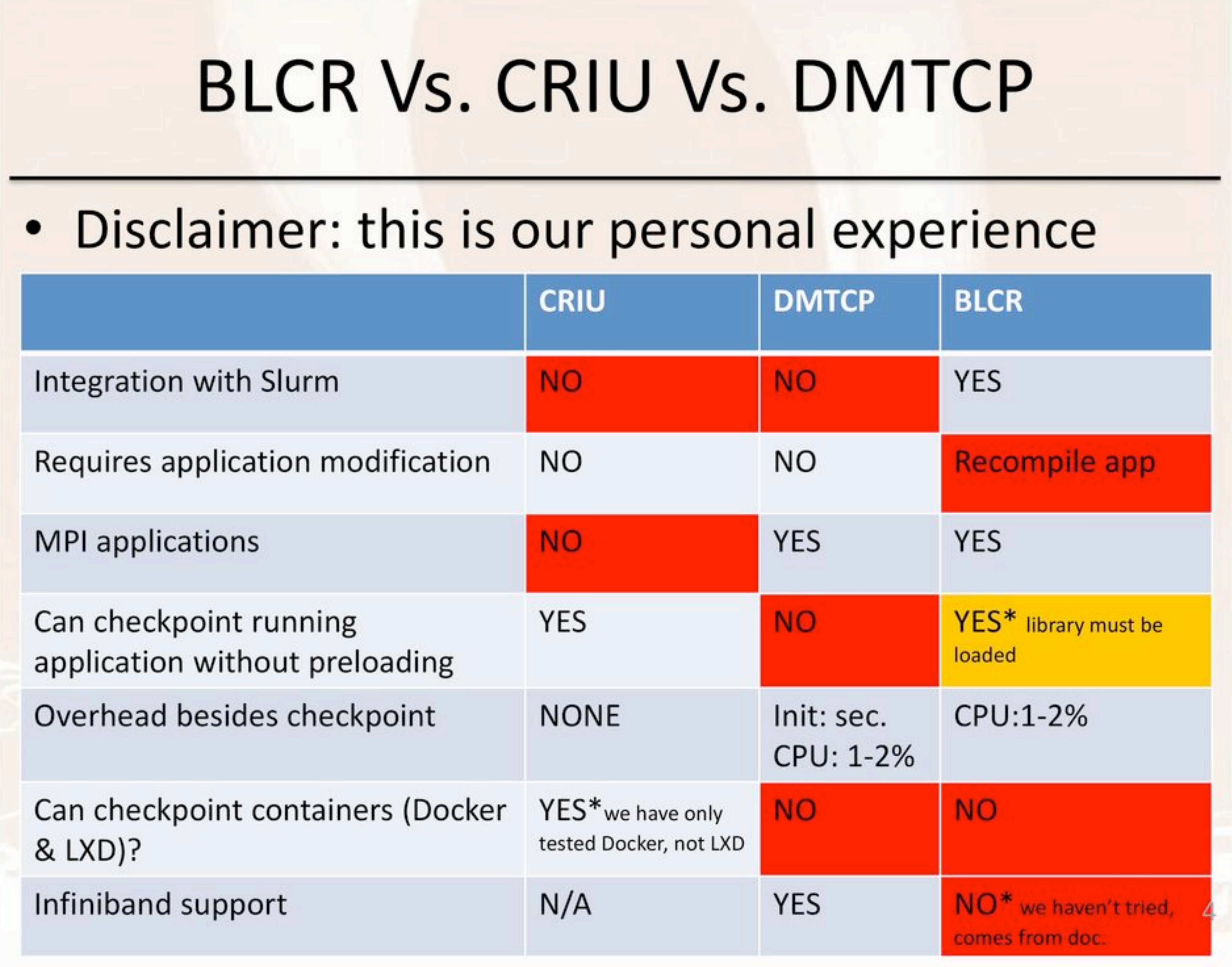
Slurm send USR1 between 1 and 2 minutes
Bash catch the message send Ctrl-c to python
python: Catch control-C to save state
Automatic resubmission

4 Making non restartable software restartable with DMTCP

NO code access



MPI
SLURM
Infiniband



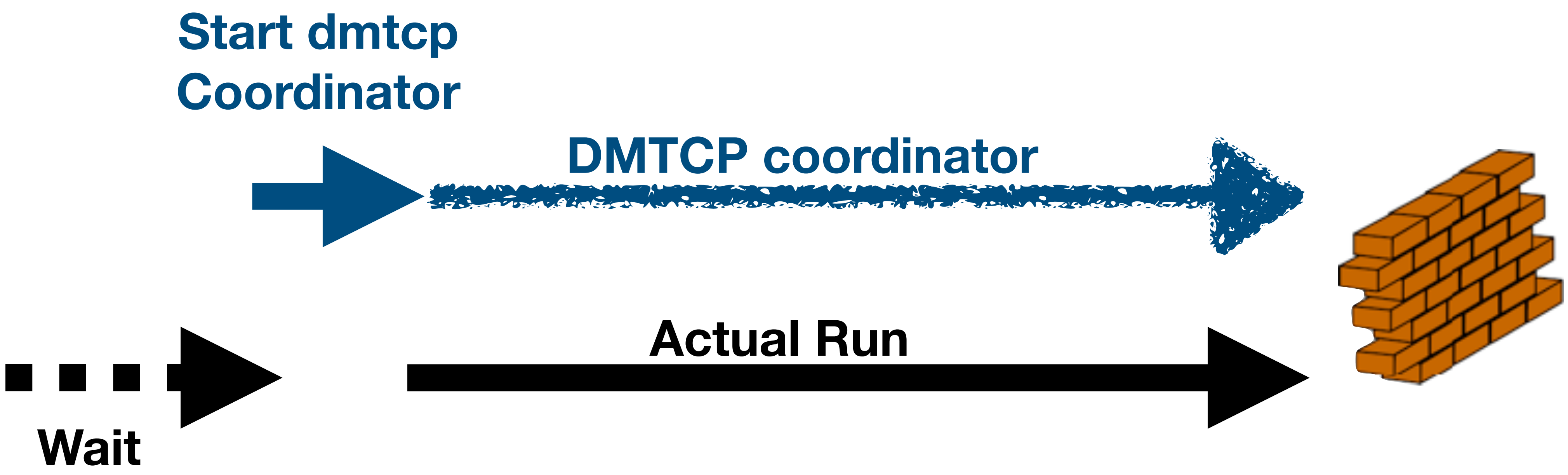
BLCR Vs. CRIU Vs. DMTCP

- Disclaimer: this is our personal experience

	CRIU	DMTCP	BLCR
Integration with Slurm	NO	NO	YES
Requires application modification	NO	NO	Recompile app
MPI applications	NO	YES	YES
Can checkpoint running application without preloading	YES	NO	YES* library must be loaded
Overhead besides checkpoint	NONE	Init: sec. CPU: 1-2%	CPU:1-2%
Can checkpoint containers (Docker & LXD)?	YES*we have only tested Docker, not LXD	NO	NO
Infiniband support	N/A	YES	NO* we haven't tried, comes from doc.

Slide from Rodriguez-Pascual

DMTCP mode



Start dmtcp Coordinator

DMTCP coordinator

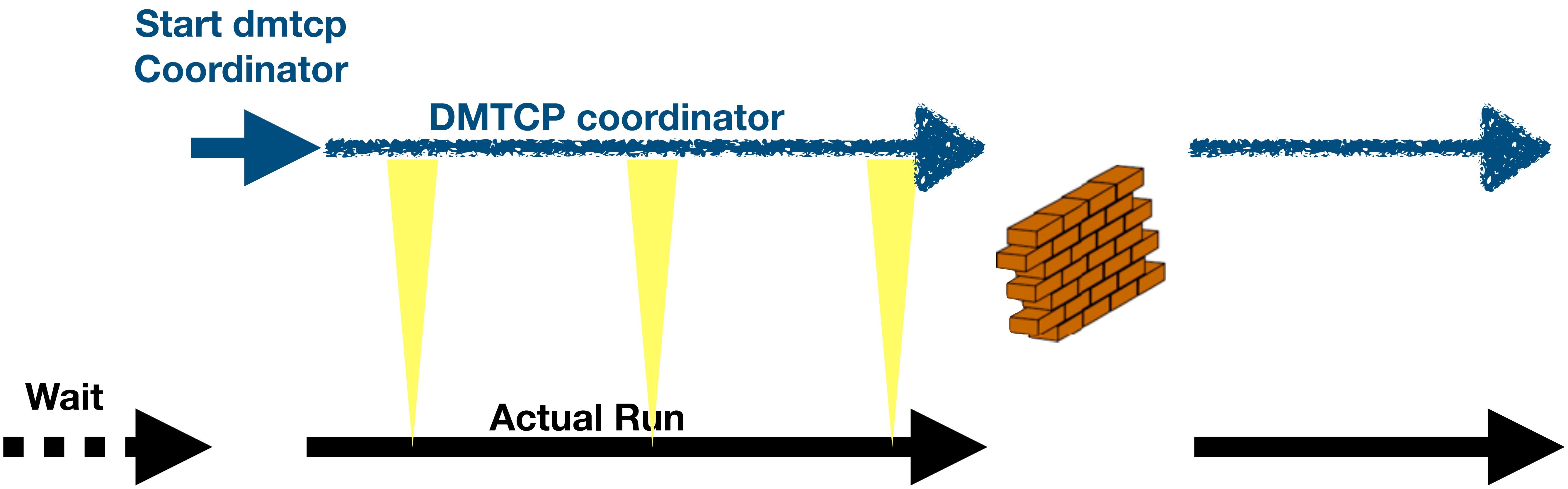
Actual Run

Wait

```
$> Module load DMTCP

$> dmtcp_launch XXX
```

DMTCP mode

Mode #1: Snapchat every X second

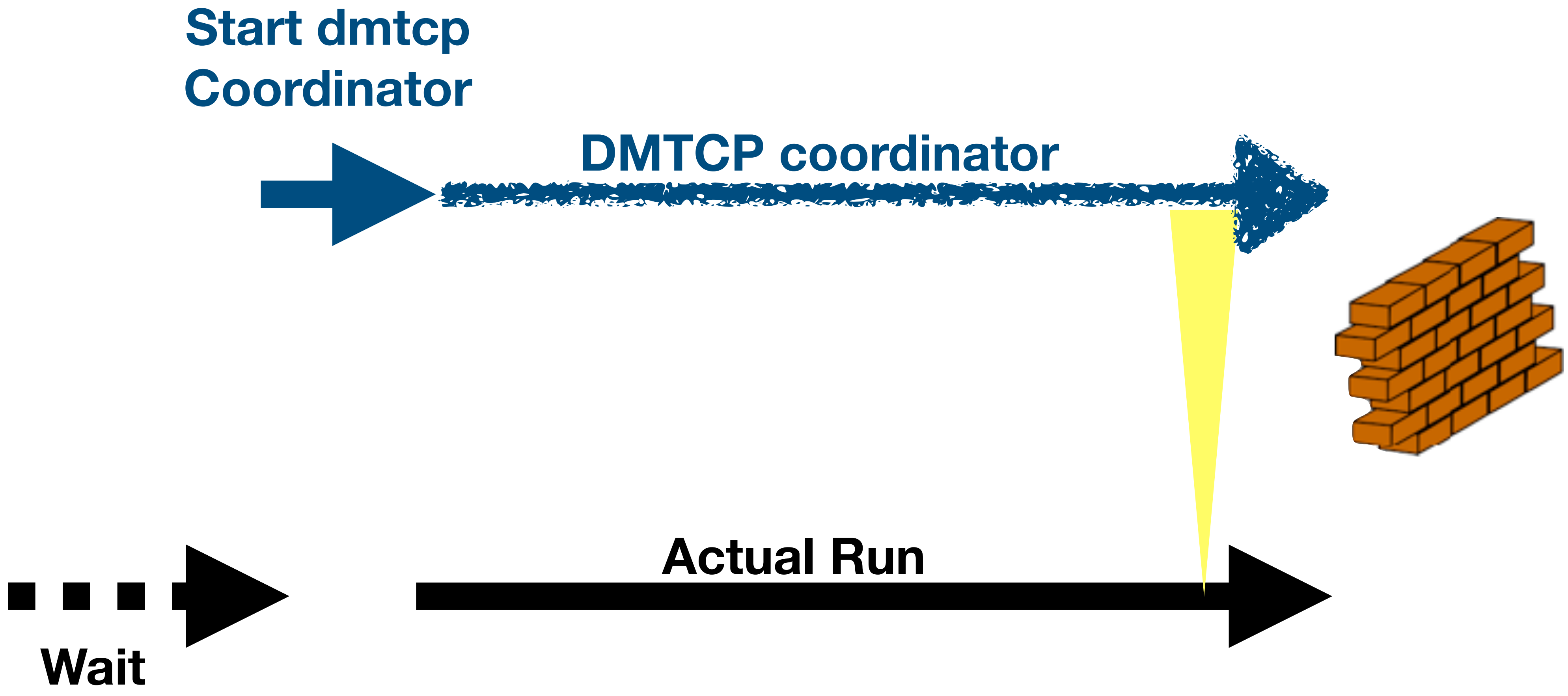


```
$> Module load DMTCP

$> dmtcp_launch XXXX

$> dmtcp_command —checkpoint
```

```
$> ./dmtcp_restart_script.sh
```

DMTCP mode

Mode #2: Snapchat on request (trigger via signal)



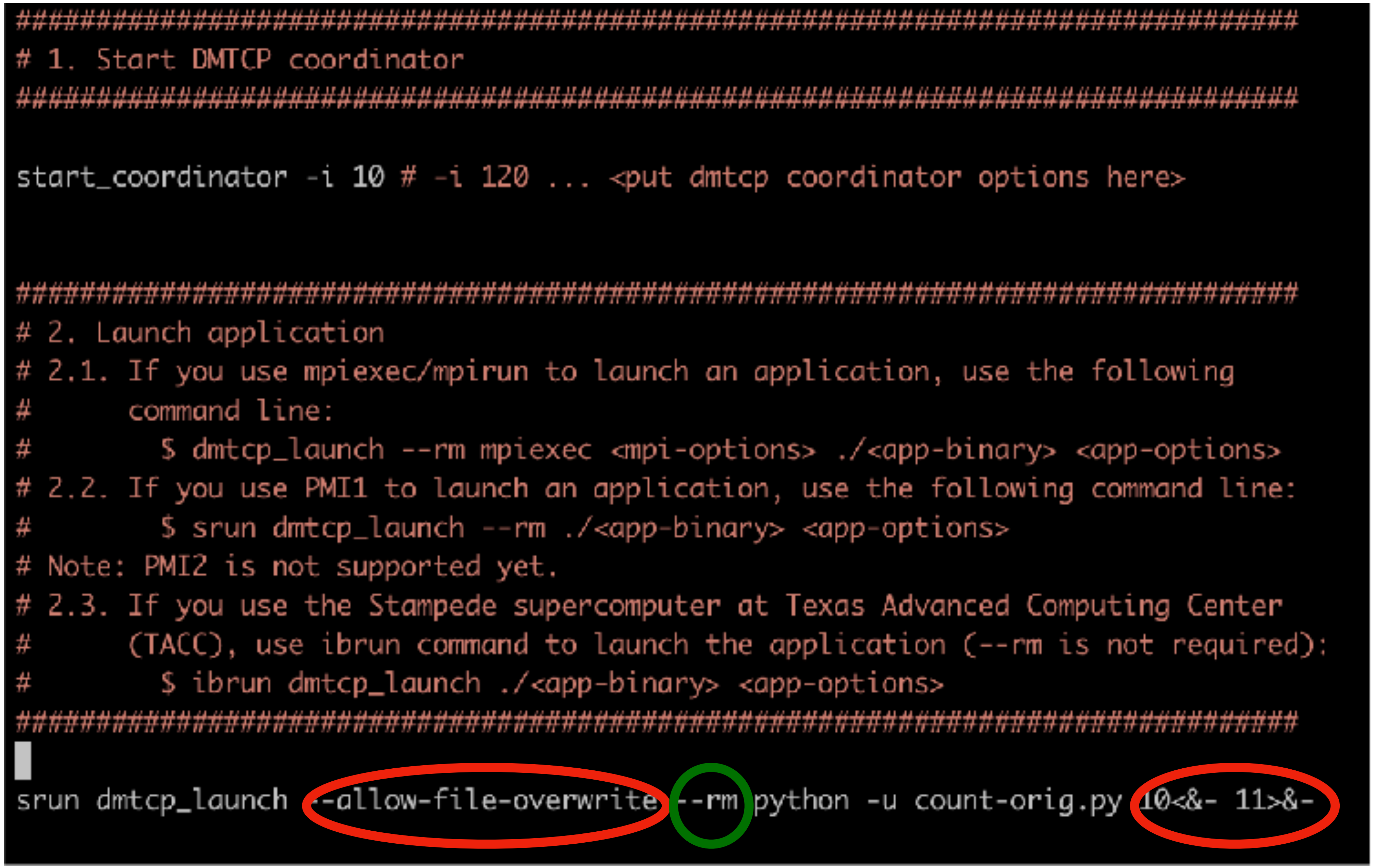
Start dmtcp Coordinator

DMTCP coordinator

Actual Run

Wait

Apply it for Slurm



```
################################################################################
# 1. Start DMTCP coordinator
################################################################################

start_coordinator -i 10 # -i 120 ... <put dmtcp coordinator options here>


################################################################################
# 2. Launch application
# 2.1. If you use mpiexec/mpirun to launch an application, use the following
#      command line:
#        $ dmtcp_launch --rm mpiexec <mpi-options> ./<app-binary> <app-options>
# 2.2. If you use PMI1 to launch an application, use the following command line:
#        $ srun dmtcp_launch --rm ./<app-binary> <app-options>
# Note: PMI2 is not supported yet.
# 2.3. If you use the Stampede supercomputer at Texas Advanced Computing Center
#      (TACC), use ibrun command to launch the application (--rm is not required):
#        $ ibrun dmtcp_launch ./<app-binary> <app-options>
################################################################################

srun dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10<&- 11>&-
```

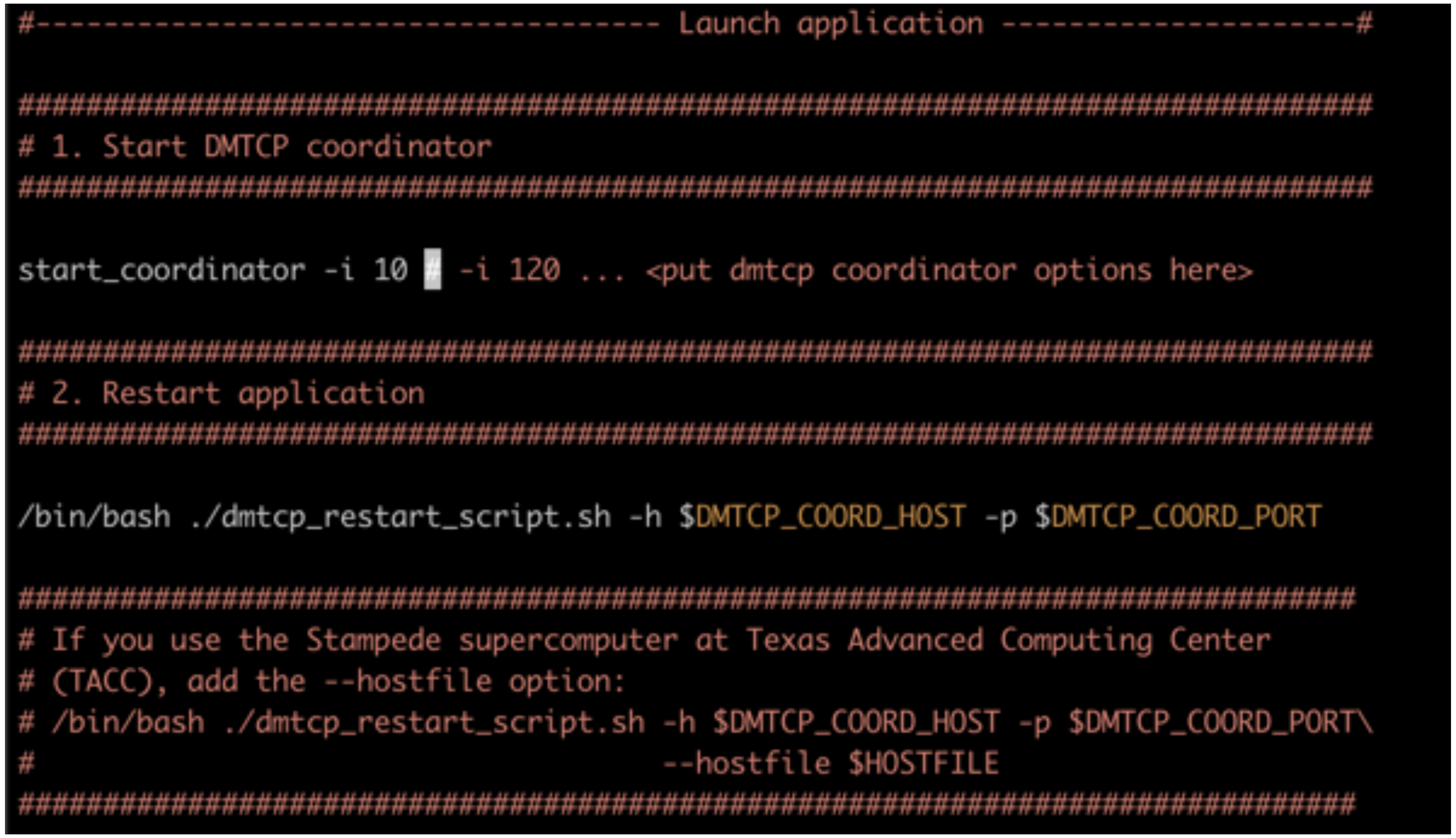
start coordinator

Normal job with decorator

Slurm aware

Lemaitre3 specific!

Resubmit



```
#----------------------------------- Launch application ---------------------#

################################################################################
# 1. Start DMTCP coordinator
################################################################################

start_coordinator -i 10 # -i 120 ... <put dmtcp coordinator options here>

################################################################################
# 2. Restart application
################################################################################

/bin/bash ./dmtcp_restart_script.sh -h $DMTCP_COORD_HOST -p $DMTCP_COORD_PORT

###############################################################################
# If you use the Stampede supercomputer at Texas Advanced Computing Center
# (TACC), add the --hostfile option:
# /bin/bash ./dmtcp_restart_script.sh -h $DMTCP_COORD_HOST -p $DMTCP_COORD_PORT\
#                                     --hostfile $HOSTFILE
###############################################################################
```

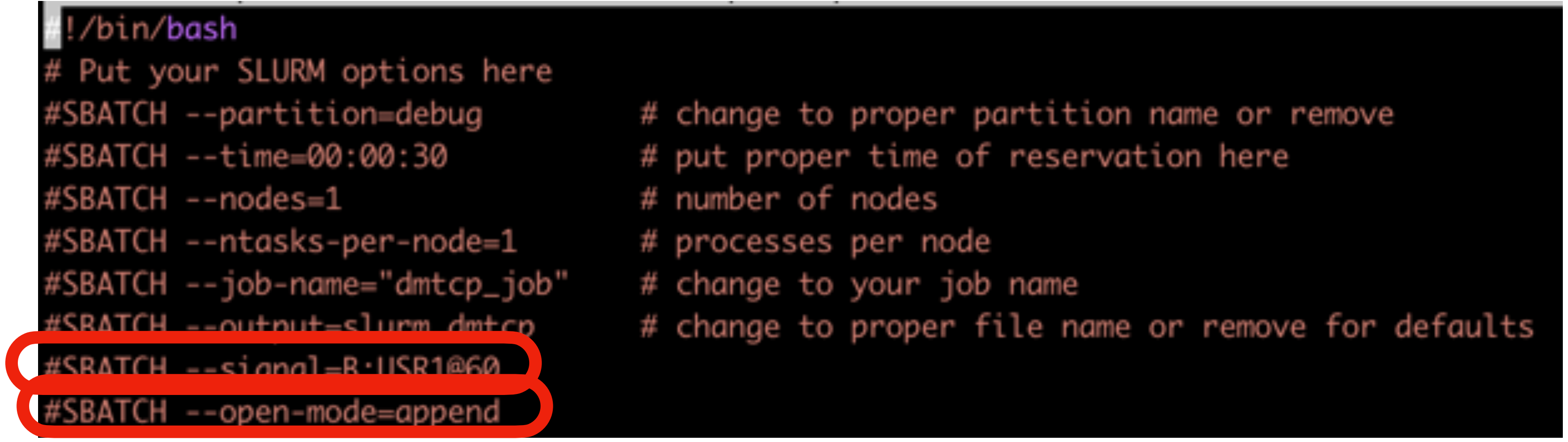
start coordinator

Script created by
previous run

Let’s combine everything

Use DMTCP with periodic check
add an additional checkpoint before wall time
Auto resubmit

Solution

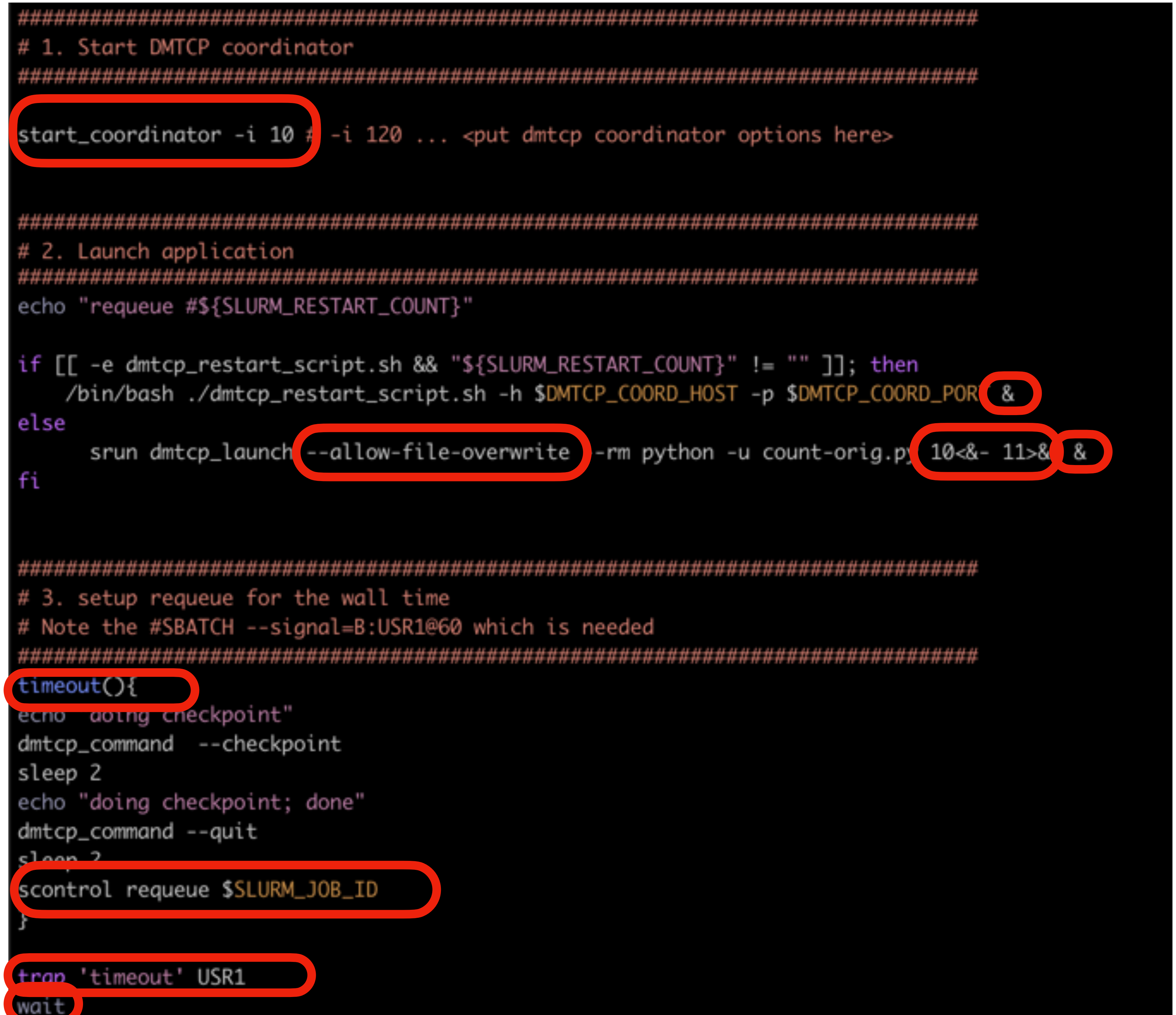


```
#!/bin/bash
# Put your SLURM options here
#SBATCH --partition=debug            # change to proper partition name or remove
#SBATCH --time=00:00:30              # put proper time of reservation here
#SBATCH --nodes=1                    # number of nodes
#SBATCH --ntasks-per-node=1          # processes per node
#SBATCH --job-name="dmtcp_job"       # change to your job name
#SBATCH --output=slurm_dmtcp         # change to proper file name or remove for defaults
#SBATCH --signal=B:USR1@60
#SBATCH --open-mode=append
```

Periodic checkpoint

Checkpoint at walltime

Auto-resubmit



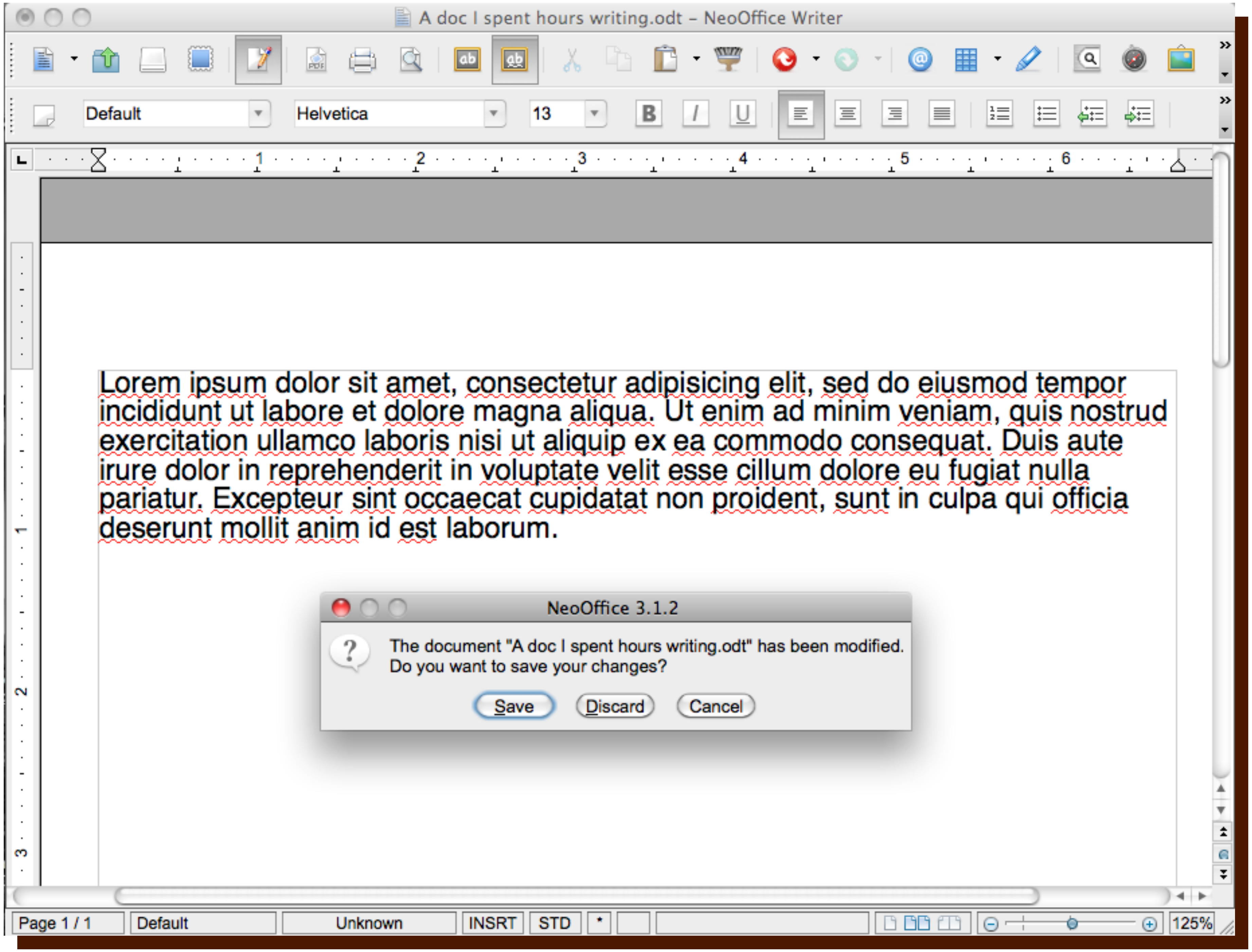
```
################################################################################
# 1. Start DMTCP coordinator
################################################################################

start_coordinator -i 10 # -i 120 ... <put dmtcp coordinator options here>


################################################################################
# 2. Launch application
################################################################################
echo "requeue #${SLURM_RESTART_COUNT}"

if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    /bin/bash ./dmtcp_restart_script.sh -h $DMTCP_COORD_HOST -p $DMTCP_COORD_PORT &
else
      srun dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10<&- 11>&- &
fi


################################################################################
# 3. setup requeue for the wall time
# Note the #SBATCH --signal=B:USR1@60 which is needed
################################################################################
timeout(){
echo "doing checkpoint"
dmtcp_command  --checkpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```

Demo #4

slurm_dmtcp_solution.sub

Summary, Wrap-up and Conclusions.

Never click 'Discard' again...



A doc I spent hours writing.odt – NeoOffice Writer

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

NeoOffice 3.1.2

The document "A doc I spent hours writing.odt" has been modified. Do you want to save your changes?