



Checkpointing

Olivier Mattelaer
CP3/CISM



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:

With checkpointing:

Checkpointing:

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

Today's agenda:

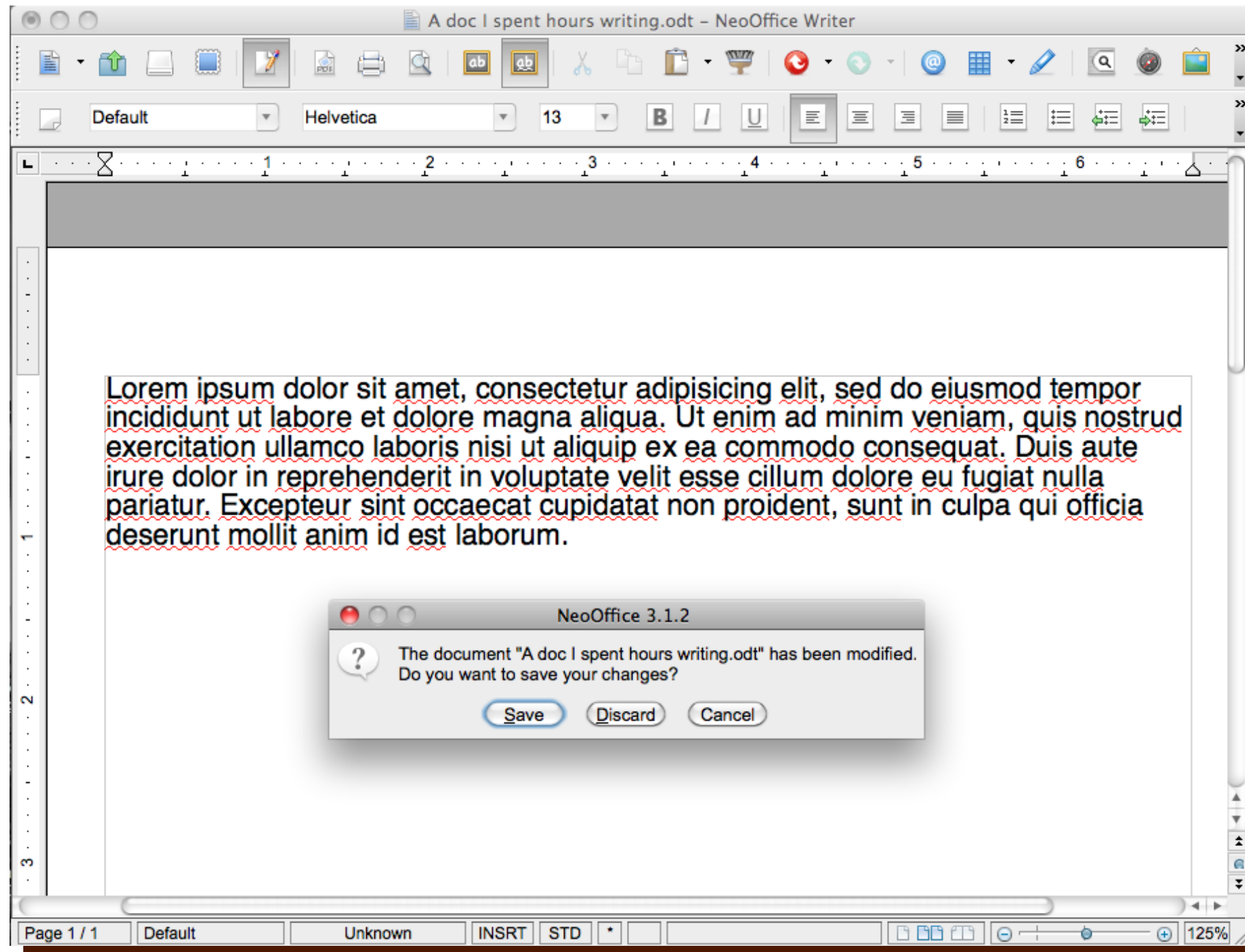
1. General concepts and scientific soft.
2. Working with Signals
3. Slurm recipes
4. DMTCP



Why do we need
checkpointing



Imagine a text editor without 'checkpointing' ...



The idea:

Save the program state

every time a checkpoint is encountered

and restart from there upon (un)planned stop

rather than bootstrap again from scratch

Values in variables
Open files
...

Position in the code
Signal or event
...

starting loops at iteration 0
creating tmp files
...

Goals of checkpointing in HPC:

1. Fit in time constraints
2. Debugging, monitoring
3. Cope with hardware failures
4. Job preemption

Goals of checkpointing in HPC:

1. Fit in time constraints

All clusters limit maximum 'wall' time of jobs
to allow high job turnover

to ensure fair time sharing of the cluster

(and reduce waiting times...)

Goals of checkpointing in HPC:

2. Debugging, monitoring

Checkpointing means saving the state on disk

-> You can view the state while the job is running

-> You can restart at the checkpoint before a bug occurred

Goals of checkpointing in HPC:

3. Cope with hardware failures

447	storage04 inaccessible	storage04 -	■ Clos	2016-01-29 15:20	
446	HDD failure	storage04 -	■ Clos	2015-11-10 10:19	Remplacement de disque dur
445	HDD failure 2TB disks (led orange mais statut ok?)	vmhp0 -	■ Clos	2015-11-10 10:14	Remplacement de disque dur
444	HDD failure slot 0	hmem -	■ Clos	2015-11-10 10:09	Remplacement de disque dur
443	HDD failure	ImP2k1a -	■ Clos	2015-11-05 14:18	Remplacement de disque dur
442	HDD failure	ImP2k1a -	■ Clos	2015-10-12 09:38	Remplacement de disque dur
441	Serveur hors tension. Pas moyen de le redemarrer	ImWn087 -	■ Clos	2015-09-21 14:49	Remplacement carte mère
440	HD Failure	vmhp0 -	■ Clos	2015-09-17 14:57	Remplacement de disque dur
439	HD Failure	hmem02 -	■ Clos	2015-09-17 14:24	Remplacement de disque dur
438	Disk Failure	storage05 -	■ Clos	2015-09-01 16:42	Remplacement de disque dur
437	HDD Failure	storage05 -	■ Clos	2015-08-27 14:51	Remplacement de disque dur
436	HD Failure	mback145 -	■ Clos	2015-08-27 10:28	Reboot
435	CPU IERR error	mback145 -	■ Clos	2015-08-24 09:48	Reboot
434	CPU IERR error	node006 -	■ Clos	2015-08-10 17:00	Reboot
433	ECC warning on DIMM2 (Cleared)	mback052 -	■ Clos	2015-08-10 16:58	Reset du System Event Log
432	Power failure	ImWn046 -	■ Clos	2015-07-27 10:04	Remplacement carte mère
431	Fan failure FAN 2	mback106 -	■ Clos	2015-07-27 10:03	Remplacement de ventilateur
430	Fan failure FAN 5	mback114 -	■ Clos	2015-07-27 10:02	Remplacement de ventilateur
429	Disk degraded	mback163 -	■ Clos	2015-07-27 10:00	Remplacement de disque dur
428	Memory detected but is not configurable	mback115 -	■ Clos	2015-07-17 16:22	Décomission
427	HD Failure	hmem02 -	■ Clos	2015-07-16 15:31	Remplacement de disque dur
426	ECC warning on DIMM8 (Cleared)	mback056 -	■ Clos	2015-06-19 14:38	Reset du System Event Log
425	Node down	ImPp003 -	■ Clos	2015-06-01 15:55	Remplacement CPU
424	ECC warning on DIMM8 (Cleared)	mback056 -	■ Clos	2015-06-01 08:57	Reset du System Event Log
423	Memory detected but is not configurable	node009 -	■ Clos	2015-05-28 08:48	Décomission
422	ECC warning on DIMM6 (Cleared)	mback210 -	■ Clos	2015-05-28 08:46	Reset du System Event Log
421	unexpected shutdown	ImWn086 -	■ Clos	2015-04-10 10:00	Reboot
420	ECC error DIMM B2	hmem04 -	■ Clos	2015-04-09 13:40	Reset du System Event Log
419	Power status off	ImWn072 -	■ Clos	2015-04-08 14:37	Remplacement carte mère
418	Cannot restart the server	ImWn009 -	■ Clos	2015-04-08 14:34	Remplacement carte mère
417	ECC warning on DIMM7 (Cleared)	node009 -	■ Clos	2015-03-17 16:50	Reset du System Event Log
416	Node down	mback216 -	■ Clos	2015-03-16 11:46	Reboot
415	ECC warning on DIMM1 (Cleared)	node009 -	■ Clos	2015-03-09 15:50	Reset du System Event Log
414	HD Failure	node010 -	■ Clos	2015-02-16 10:50	Remplacement de disque dur
413	HD Failure	node010 -	■ Clos	2015-02-12 08:48	Remplacement de disque dur
412	Memory failure detected	node004 -	■ Clos	2015-02-09 14:47	Reboot
411	ecc warning on DIMM3 (Cleared)	mback149 -	■ Clos	2015-02-09 14:41	Reset du System Event Log
410	Node down	node002 -	■ Clos	2015-02-05 09:48	Remplacement de DIMM
409	server power status off	mback151 -	■ Clos	2015-02-05 09:15	Changement de câble
408	Node down	mback035 -	■ Clos	2015-02-02 10:17	
407	ECC warning on DIMM5 (Cleared)	mback197 -	■ Clos	2015-01-19 09:10	Reset du System Event Log
406	unexpected shutdown	ImWn079 -	■ Clos	2015-01-06 16:44	

Goals of checkpointing in HPC:

4. Job preemption

Not used at CÉCI, preemption is the ability for a high-priority job to re-queue a low-priority job



1

**Checkpointing with scientific software
Do they support checkpointing?**

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

Working with checkpoint-restart-able software



Gaussian 09 Frequently Asked Question

How can I restart a job that was interrupted?

Many Gaussian jobs that are stopped prematurely — e.g., due to a machine crash, a power failure, manually killing the job — can be restarted. These include geometry optimizations, frequency calculations, and CCSD and EOM-CCSD calculations. The technique to restart the jobs varies depending on the type of job. This FAQ will discuss some common cases.

Be aware that all restarts require the checkpoint file from the previous job. Some job types also require the read-write file. If the required file(s) have been deleted, then the job cannot be restarted.

http://www.gaussian.com/g_blog/faq2.htm

Working with checkpoint-restart-able software



Open Source
Molecular Dynamics

[Log In](#) [Register](#)

[Recent Changes](#) [Media Manager](#) [Sitemap](#)

Trace: • [restarting](#)

CP2K

- [About](#)
- [Science](#)
- [Features](#)
- [Download](#)
- [News , Twitter](#)
- [Positions](#)
- [Performance](#)
- [Version History](#)
- [Foundation](#)

For Users

- [Reference Manual](#)
- [Forum](#)
- [HOWTOs](#)
- [Exercises](#)
- [Acronyms](#)
- [FAQs](#)
- [Tools](#)

restarting



This text is probably out of date and needs to be revised.

Restarting

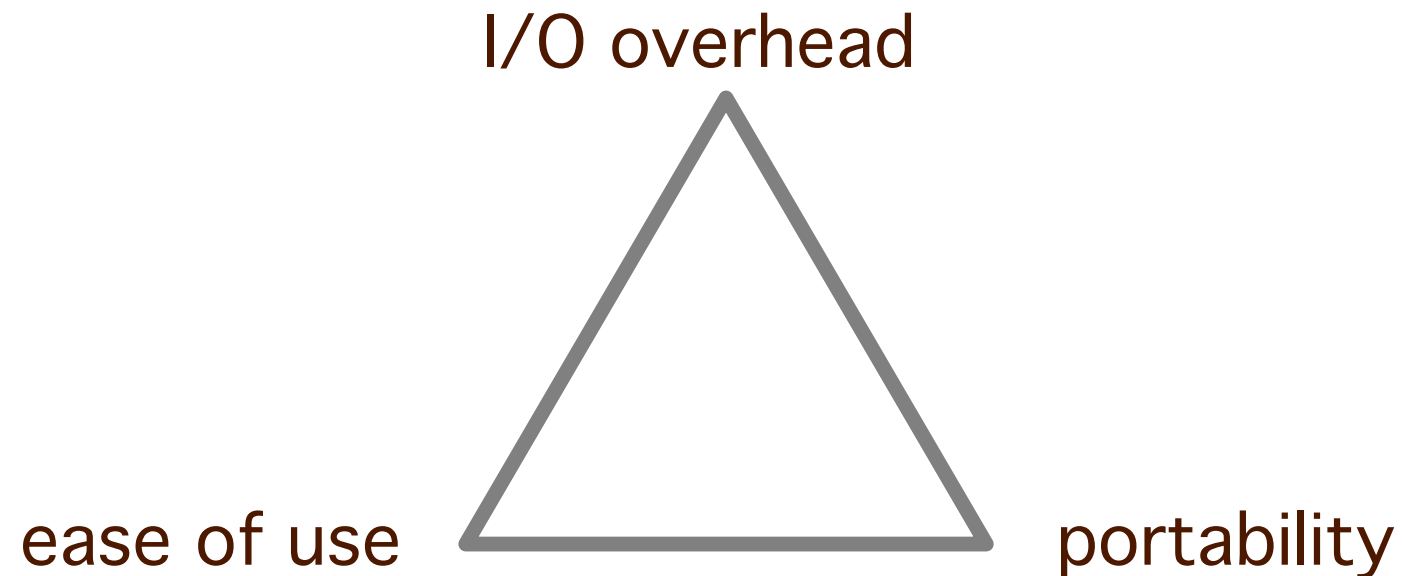
The files to restart a trajectory in cp2k are just **input files** with all the defaults made explicit. So it should not be so difficult to understand them.

Often one wants to continue a MD trajectory with slightly different parameters, for example, an equilibration run followed by a NVE run. This is done with the use of a restart file, however it can be difficult to edit or use this file directly. To make this easier one can prepare an input that takes part of its values from another input. In particular it might take positions and velocities from the restart file.

<https://www.cp2k.org/restarting>

Need to implement it yourself?

Evaluate the options : tradeoff between



Transparency for
developer

Portability to
other systems

Size of state to
save

Checkpointing
overhead

Do I need to
write a lot of
additional
code ?

Can I stop on
one system and
restart on
another ?

How many GB
of disk does it
require ?

How many
FLOPs lost to
ensure
checkpointing ?

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

^C

^D

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

^Z

— kill -9

— kill

— fg, bg

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

e.g. →
→

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' with an associated default action

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

Demo #2

`count-signal.py`

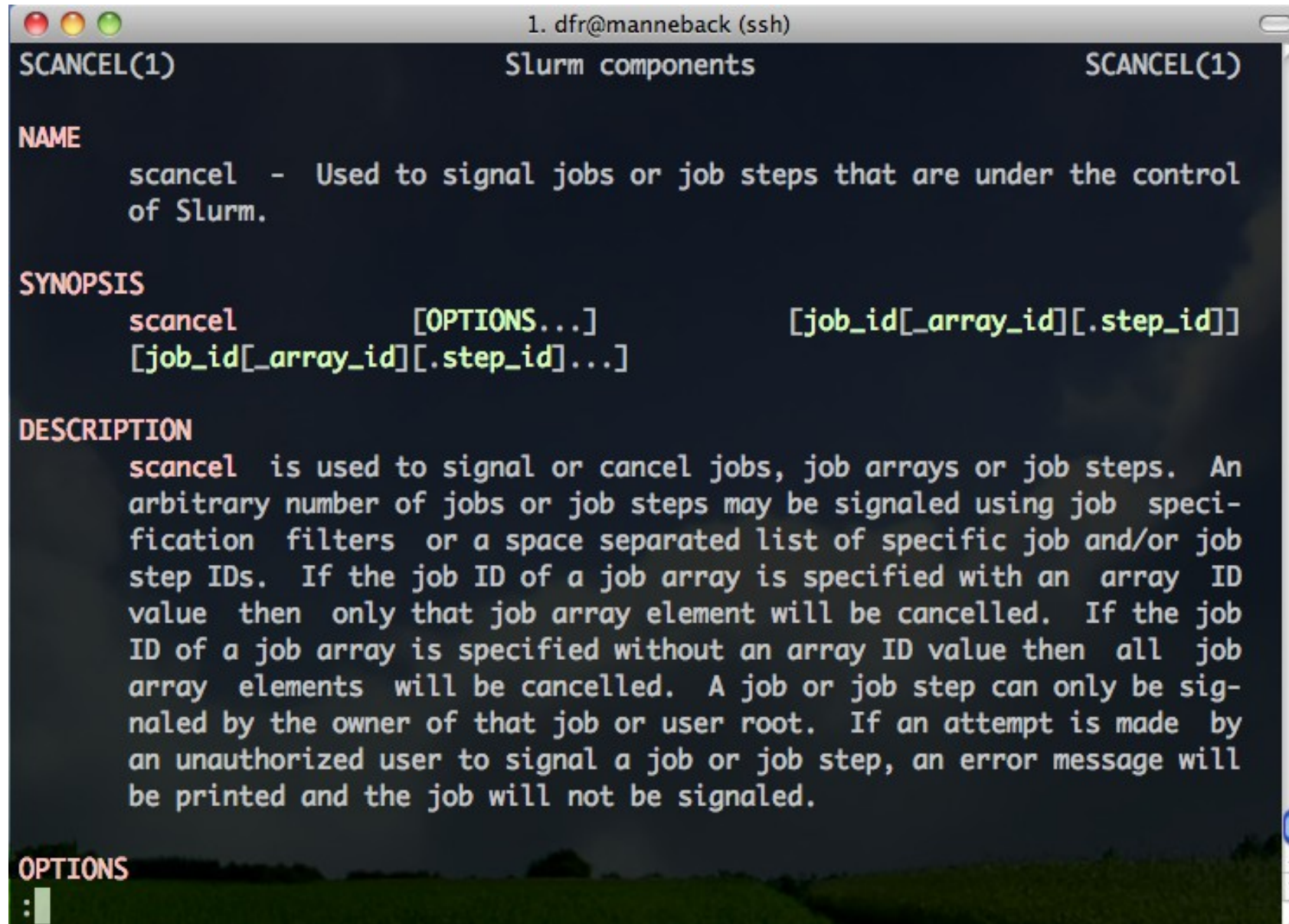
Catch control-C to save state



3

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

scancel is used to send signals to jobs



A terminal window titled "1. dfr@manneback (ssh)" with a dark background. The window displays the man page for the `scancel` command. The page is titled "SCANCEL(1) Slurm components SCANCEL(1)". It includes sections for NAME, SYNOPSIS, DESCRIPTION, and OPTIONS. The NAME section states that `scancel` is used to signal jobs or job steps under Slurm control. The SYNOPSIS section shows the command syntax: `scancel [OPTIONS...] [job_id[_array_id][.step_id]] [job_id[_array_id][.step_id]...]`. The DESCRIPTION section explains that `scancel` is used to signal or cancel jobs, job arrays, or job steps, and that it can be used with job specification filters or a list of specific IDs. The OPTIONS section is partially visible at the bottom.

```
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

```
root@lm3-m001:~ (ssh)
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

```
root@lm3-m001:~ (ssh)
File Edit Options Buffers Tools Sh-Script Help
#!/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

Demo #3

`submit-signal.sh`

python: Catch control-C to save state

Slurm send control-C between 1 and 2 minutes

`submit-signal2.sh`

python: Catch control-C to save state

Slurm send control-C between 1 and 2 minutes

Automatic re-queuing

Adding requeuing automatically

```
root@lm3-m001:~ (ssh)
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash

#SBATCH --job-name=test
#SBATCH --output=test.signal.watch
#SBATCH --open-mode=append
#SBATCH --time=0-00:05:00
#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 #4

`slurm-signal-requeue.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

Or chain the jobs...



slurm
workload manager
Version 2.6

About

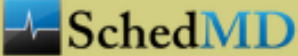
- [Overview](#)
- [Meetings](#)
- [What's New](#)
- [Publications](#)
- [Testimonials](#)
- [SLURM Team](#)

Using

- [Tutorials](#)
- [Documentation](#)
- [FAQ](#)
- [Getting Help](#)
- [Mailing Lists](#)

Installing

- [Download](#)
- [Installation Guide](#)
- [Platforms](#)



SchedMD

-d, --dependency=<dependency_list>

Defer the start of this job until the specified dependencies have been satisfied completed. <dependency_list> is of the form <type:job_id[:job_id] [,type:job_id[:job_id]]>. Many jobs can share the same dependency and these jobs may even belong to different users. The value may be changed after job submission using the scontrol command.

after:job_id[:jobid...]

This job can begin execution after the specified jobs have begun execution.

afterany:job_id[:jobid...]

This job can begin execution after the specified jobs have terminated.

afternotok:job_id[:jobid...]

This job can begin execution after the specified jobs have terminated in some failed state (non-zero exit code, node failure, timed out, etc).

afterok:job_id[:jobid...]

This job can begin execution after the specified jobs have successfully executed (ran to completion with an exit code of zero).

expand:job_id

Resources allocated to this job should be used to expand the specified job. The job to expand must share the same QOS (Quality of Service) and partition. Gang scheduling of resources in the partition is also not supported.

singleton

This job can begin execution after any previously launched jobs sharing the same job name and user have terminated.

-D, --workdir=<directory>

Set the working directory of the batch script to *directory* before it is executed.



4

Making non restartable software
restartable with DMTCP

DMTCP: Distributed MultiThreaded CheckPointing

About DMTCP:

DMTCP (Distributed MultiThreaded Checkpointing) is a tool to transparently checkpoint the state of multiple simultaneous applications, including multi-threaded and distributed applications. It operates directly on the user binary executable, without any Linux kernel modules or other kernel modifications.

Among the applications supported by DMTCP are Open MPI, MATLAB, Python, Perl, and many programming languages and shell scripting languages. Starting with release 1.2.0, DMTCP also supports [GNU screen](#) sessions, including vim/cscope and emacs. With the use of TightVNC, it can also checkpoint and restart X Window applications, as long as they do not use extensions (e.g.: no OpenGL, no video). See the [QUICK-START](#) file for further details.

DMTCP supports InfiniBand internally as of Aug., 2013, and will soon be released.

DMTCP is also the basis for [URDB, the Universal Reversible Debugger](#). URDB was an experimental project for reversibility for four debuggers: gdb, MATLAB, python (pdb), and perl (perl -d). It is now obsolete, and work is continuing on a newer internal project, which will be released as open source in the future.

[News](#) | [See Also](#) | [Authors](#) | [Acknowledgement](#)

Announcement!

We are currently looking for well qualified applicants who are interested in joining a Ph.D. program in order to do research on checkpointing and reversible debugging. Interested applicants should write to Gene Cooperman (gene@ccs.neu.edu) at Northeastern University.

[Home](#)

[Downloads](#)

[SF project page](#)

[Browse Source](#)

[Demo](#)

[Supported Apps](#)

[Condor Integration](#)

[Manual/Documentation](#)

[API](#)

[FAQ](#)

[Publications](#)

[Contact Us](#)

Advertised Features

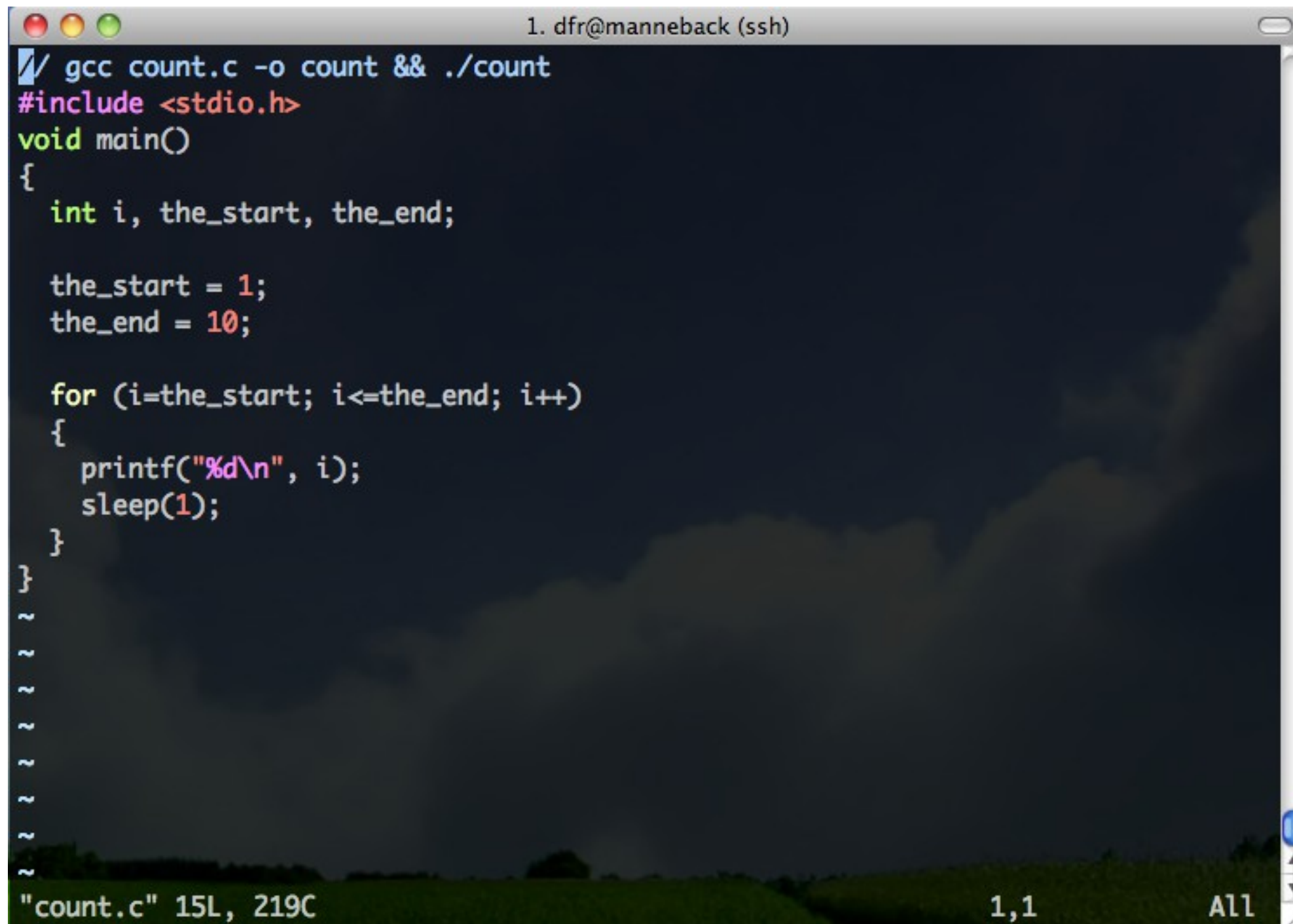
- Distributed Multi-Threaded CheckPointing
- Works with Linux Kernel 2.6.9 and later
- Supports sequential and multi-threaded computations across single/multiple hosts
- Entirely in user space (no kernel modules or root privilege)
- Transparent (no recompiling, no re-linking)
- Written at Northeastern U. and MIT and under active development for 5+ years
- LGPL'd and freely available
- No remote I/O
- Supports threads, mutexes/semaphores, forks, shared memory, exec, and many more

From their FAQ:

“ **What types of programs can DMTCP checkpoint?**

It checkpoints most binary programs on most Linux distributions. Some examples on which users have verified that DMTCP works are: Matlab, R, Java, Python, Perl, Ruby, PHP, Ocaml, GCL (GNU Common Lisp), emacs, vi/cscope, Open MPI, MPICH-2, OpenMP, and Cilk. See [Supported Applications](#) for further details. Our goal is to support DMTCP for all vanilla programs. **If DMTCP does not work correctly** on your program, **then this is a bug in DMTCP.** We would be appreciative if you can then [file a bug report with DMTCP.](#) ”

Imagine a non-checkpointable program



```
1. dfr@manneback (ssh)
// gcc count.c -o count && ./count
#include <stdio.h>
void main()
{
    int i, the_start, the_end;

    the_start = 1;
    the_end = 10;

    for (i=the_start; i<=the_end; i++)
    {
        printf("%d\n", i);
        sleep(1);
    }
}
~
~
~
~
~
~
~
~
~
~
"count.c" 15L, 219C 1,1 All
```

Run with dmtcp_launch (runs monitoring daemon if necessary)

```
1. dfr@leleve (ssh)
dfr@leleve:~/Checkpointing $ dmtcp_launch ./count & sleep 4 ; dmtcp_command --quiet
--checkpoint ; sleep 1 ; dmtcp_command --quiet --quit
[1] 2976
dmtcp_launch (DMTCP + MTCP) version 2.0
Copyright (C) 2006-2013 Jason Ansel, Michael Rieker, Kapil Arya, and
Gene Cooperman

This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; see COPYING file for details.
(Use flag "-q" to hide this message.)

dmtcp_coordinator starting...
  Host: leleve.cism.ucl.ac.be (0.0.0.0)
  Port: 7779
  Checkpoint Interval: disabled (checkpoint manually instead)
  Exit on last client: 1
Backgrounding...
1
2
3
4
5
[1]+  Done                  dmtcp_launch ./count
dfr@leleve:~/Checkpointing $ ls -rtl|tail -1
-rwxrw-r--  1 dfr dfr    5167 Oct 15 11:51 dmtcp_restart_script_1dcda56f5a2723b6-40000-
525d1005.sh
dfr@leleve:~/Checkpointing $
```


Restart with dmtcp_restart_script.sh

```
1. dfr@leleve (ssh)
5
[1]+  Done                  dmtcp_launch ./count
dfr@leleve:~/Checkpointing $ ls -rtl|tail -1
-rwxr--r-- 1 dfr dfr  5167 Oct 15 11:52 dmtcp_restart_script_1dcda56f5a2723b6-40000-525d1043.sh
dfr@leleve:~/Checkpointing $ ./dmtcp_restart_script.sh
dmtcp_restart (DMTCP + MTCP) version 2.0
Copyright (C) 2006-2013 Jason Ansel, Michael Rieker, Kapil Arya, and
                        Gene Cooperman

This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; see COPYING file for details.
(Use flag "-q" to hide this message.)

dmtcp_coordinator starting...
  Host: leleve.cism.ucl.ac.be (0.0.0.0)
  Port: 7779
  Checkpoint Interval: disabled (checkpoint manually instead)
  Exit on last client: 1
Backgrounding...
5
6
7
8
9
10
^C
dfr@leleve:~/Checkpointing $
```

Launch the coordinator and the program with automatic checkpointing every 30 seconds

```
root@lm3-m001:~  
File Edit Options Buffers Tools Sh-Script Help  
#!/bin/sh  
# Sample SLURM batch script to run a program  
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.  
# Be sure to also modify the dmtcp_launch line for your actual job.  
#SBATCH --partition=debug  
#SBATCH --ntasks=1  
#SBATCH --time=00:03:00  
#SBATCH --output=slurm.dmtcp  
  
# Report actual hostname to user.  
  
# If you install DMTCP in your user directory (not cluster-wide) you need to  
# extend the PATH variable:  
module load 2018a  
module load DMTCP  
module load Python/2.7.14-foss-2018a  
  
# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)  
srun --overcommit --ntasks=1 dmtcp_coordinator &  
# DMTCP coordinator needs to be started on localhost. Or put the other host  
# in the '-h' option.  
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).  
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup  
dmtcp_launch --allow-file-overwrite --interval 30 --rm python count-orig.py 10>&- 11>&-
```

Lemaitre3 specific!

Launch coordinator and restart program

```
root@lm3-m001:~  
File Edit Options Buffers Tools Sh-Script Help  
#!/bin/sh  
# Sample SLURM batch script for restart  
# You'll also need to customize the SBATCH lines below.  
# The script, ./dmtcp_restart_script.sh, will have been created for you  
#   by a checkpoint during the initial run.  
  
#SBATCH --ntasks=1  
#SBATCH --output=res  
#SBATCH --open-mode=append  
#SBATCH --partition=debug  
#SBATCH --time=00:05:00  
  
# Report actual hostname to user.  
#hostname  
  
# If you install DMTCP in your user directory (not cluster-wide), you'll  
# need to extend PATH variable:  
#export PATH=./dmtcp-2.0/bin:$PATH  
module load 2018a  
module load DMTCP  
  
# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)  
srun --overcommit dmtcp_coordinator &  
export DMTCP_HOST=`hostname`  
  
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).  
./dmtcp_restart_script.sh --interval=30
```

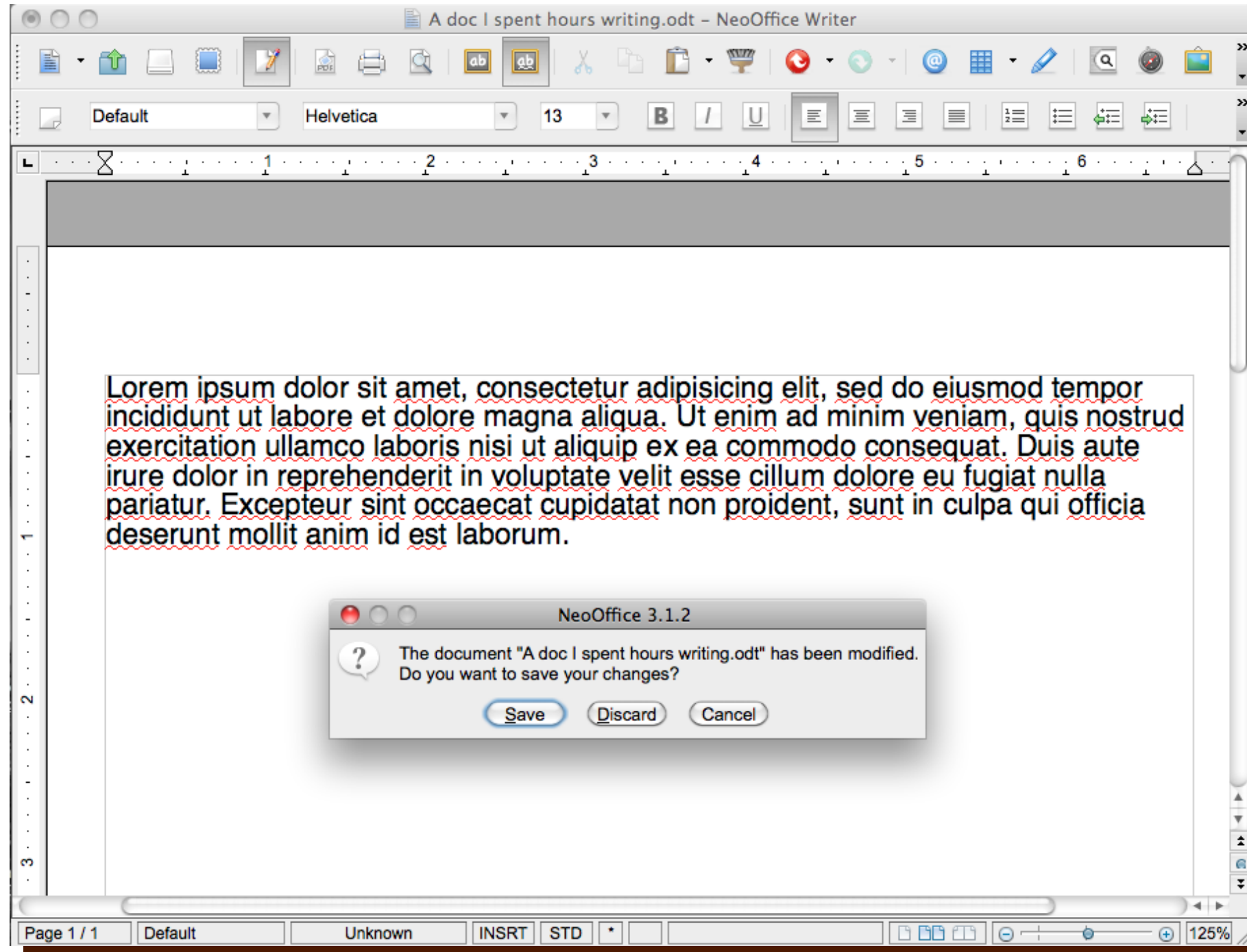
1) Coordinator

2) restart program



Summary,
Wrap-up and
Conclusions.

Never click 'Discard' again...



The submission script(s)

- Either one big one or two small ones
- Checkpoint periodically or --signal
- Requeue automatically
- Open-mode=append