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:

With checkpointing:

Checkpointing:

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

Checkpointing

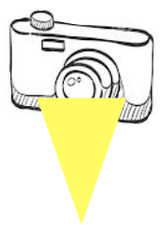
Software

- If the software has (even partial) internal checkpointing: use it !
- Typically lightweight
- At meaningful time



Hardware

- If software specific is not available (or not enough)
- Damp the RAM/... on disk
  - Heavy/slow
- Can be done any time

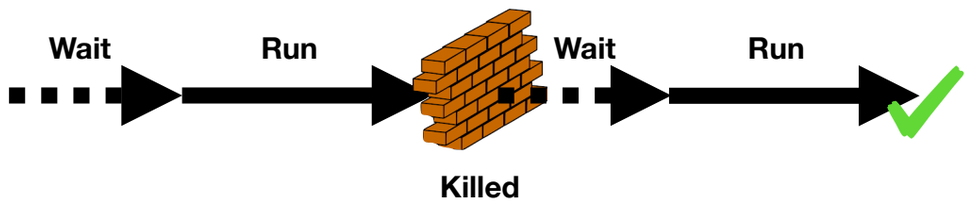




Why do we need checkpointing

?

Wall-Time



Today Agenda

- How to checkpoint every **iteration**.
  - Easy just setting the stage
- How to checkpoint **on demand**.
  - Signal
  - Every X minutes

Today Agenda

- How to checkpoint every **iteration**.
  - Easy just setting the stage

Software

- How to checkpoint **on demand**.
  - Signal
  - Every X minutes

Hardware



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
^Z

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

kill -9

kill

fg, bg

Signal handling

1. Kernel notices the signal.
2. Current execution is suspended.
3. CPU state (registers, instruction pointer, stack pointer, etc.) is saved.
4. Kernel arranges for `handler()` to run.
5. `handler()` executes on the process's normal stack.
6. When the handler returns, the saved state is restored.
7. Execution resumes where it was interrupted.

Main code interrupted/ restarted (if allowed)



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

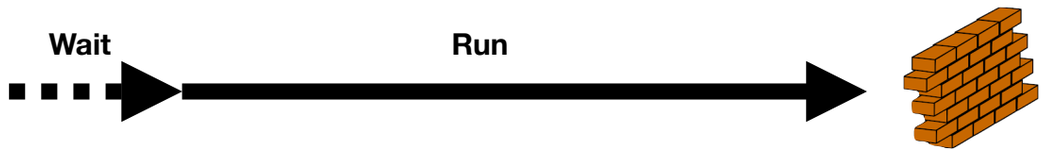
Just in time checkpointing



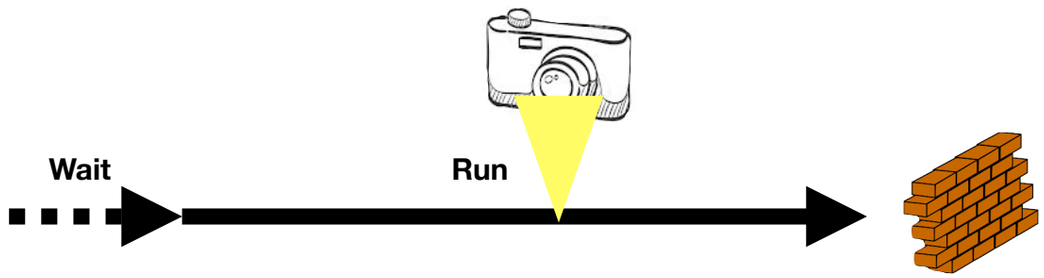


4 Making non restartable software restartable with DMTCP

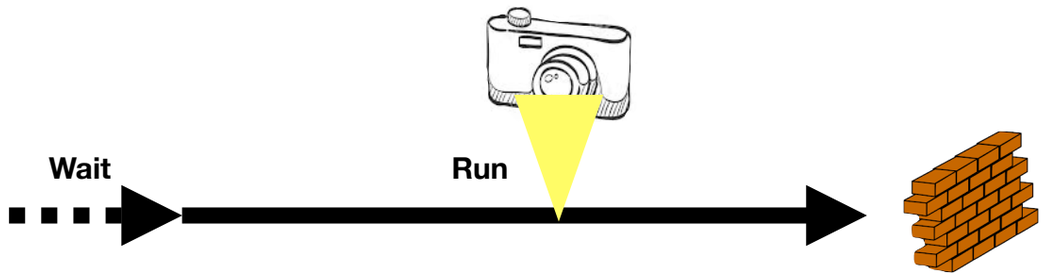
NO code access



NO code access

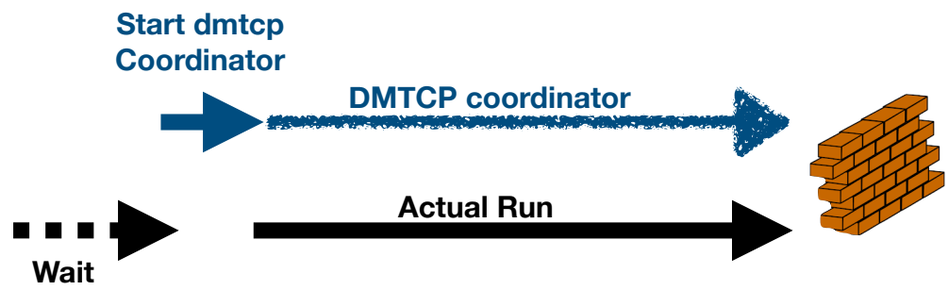


NO code access



MPI
SLURM
Infiniband

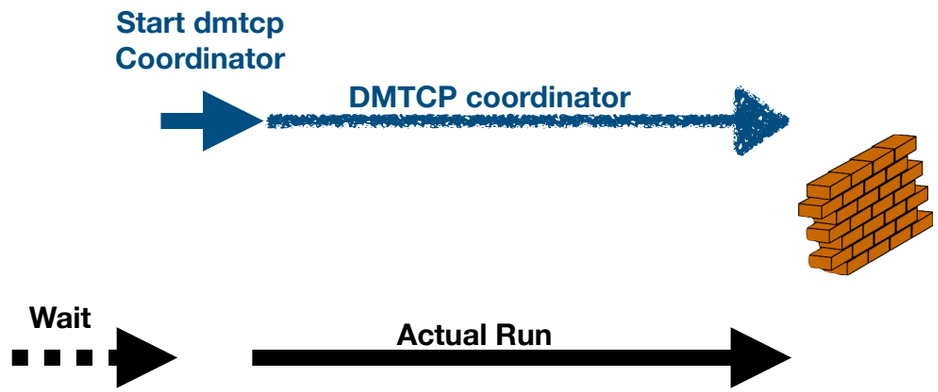
DMTCP mode




```
$> Module load DMTCP
$> dmtcp_launch XXX
```

DMTCP mode

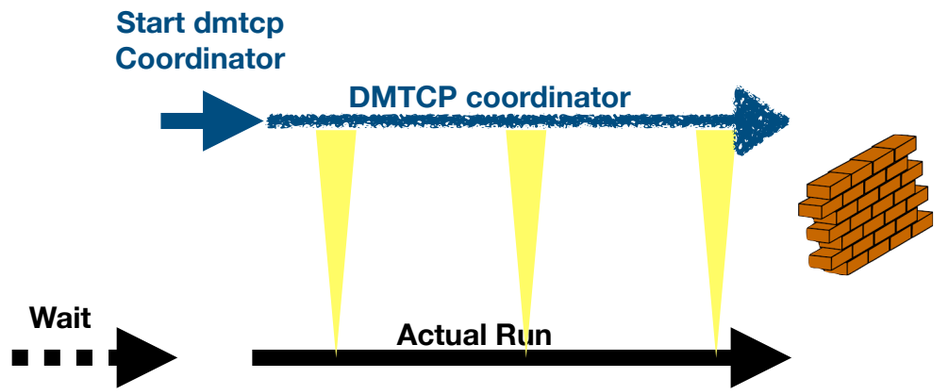
Mode #1: Snapshot every X second




```
$> Module load DMTCP
$> dmtcp_launch XXXX
$> dmtcp_command --bcheckpoint
```

DMTCP mode

Mode #1: Snapshot every X second



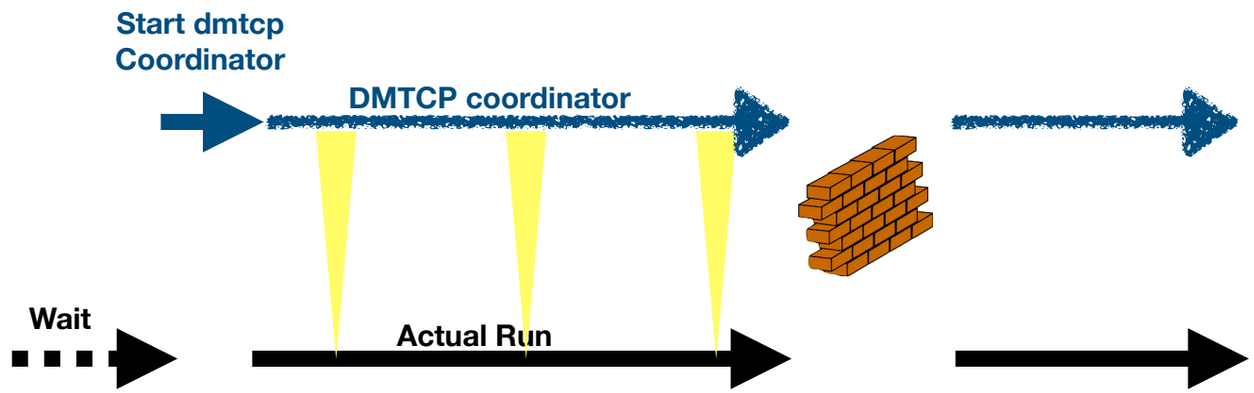

```
$> Module load DMTCP

$> dmtcp_launch XXXX

$> dmtcp_command --bcheckpoint
```

DMTCP mode

Mode #1: Snapshot every X second




```
$> Module load DMTCP

$> dmtcp_launch XXXX

$> dmtcp_command --bcheckpoint
```

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

Let’s play

Module load gameshell/checkpoint
(on lemaitre4)

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     python -u count-orig.py 10<&- 11>&-
```

start coordinator
Snapshot every 10s

Normal job with decorator

Lemaitre4 specific!

Deactivate buffering:
Slurm: stdbuf -i0 -o0 -e0 sbatch slurm_dmtcp_new.sub
Python:
You can skip buffering for a whole python process using ``python -u``

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

Deactivate buffering:
Slurm: stdbuf -i0 -o0 -e0 sbatch slurm_dmtcp_restart.sub

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
```

```
################################################################################
# 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      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 --bcheckpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```

Slurm: stdbuf -i0 -o0 -e0 sbatch slurm_dmtcp_solution.sub

Solution

Periodic checkpoint

```
#!/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
```

```
################################################################################
# 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      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 --bcheckpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```

Solution

Periodic checkpoint

Checkpoint at walltime

```
#!/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
```

```
################################################################################
# 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      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 --bcheckpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```

Solution

Periodic checkpoint

Checkpoint at walltime

```
#!/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
```

```
################################################################################
# 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      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 --bcheckpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```

Solution

Periodic checkpoint

Checkpoint at walltime

Auto-resubmit

```
#!/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

################################################################################
# 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      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 --bcheckpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```

Solution

Periodic checkpoint

Checkpoint at walltime

Auto-resubmit

```
#!/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

################################################################################
# 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     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 --bcheckpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1
wait
```



Summary, Wrap-up and Conclusions.

Never click 'Discard' again...

