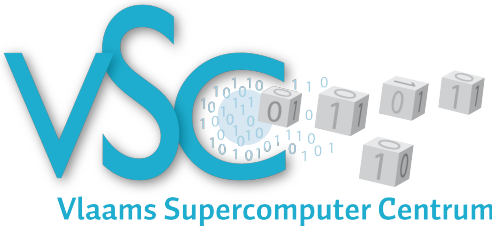
Checkpoint/Restart facilities

HPC-TNT-3 11-12-2015



Annie Cuyt ☐ Stefan Becuwe ☐ Franky Backeljauw ☐ Kurt Lust ☐ Engelbert Tijskens

Overview

- What is checkpointing?
- What problems can it solve?
- Different approaches to C/R
- Practically:
  - Applications with built-in C/R capability (native C/R)
    - VASP, LAMMPS, ...

  - csub/BLCR
  - Intel mpirun
  - dmtcp
- Case study using VASP/5.3.5-intel-2015a

What is checkpoint/restart?

- **Checkpoint =**
  - Dump the state of program in a file
- **Restart =**
  - Read the program state back in and continue the computation

What problems can it solve?

- Program did not run until completion
  - Hardware failure
  - Power problem
  - Running out of
    - Wall time
    - Memory
    - Disk space
  - Maximum wall time of a cluster exceeded
    - E.g. 3 days on current Tier-1
- Continue a job with different input parameters
  - (only for native C/R)

What problems can it solve?

- Without restart capability all cpu hours spent on the job are lost
- With restart capability only the cpu hours spent since the last checkpoint are lost

- Checkpointing adds relatively small overhead

Different approaches to C/R

Dump the state of the	
solution (procedure)	**program environment**
White box requires understanding of the state of the problem and of the solution procedure	Black box
C/R logic must be implemented (harder to maintain)	No connection to problem domain
Understandable	Unrelated to problem domain
Dumps data	Dumps bits
Smaller files	Big files (program logic is also in memory)
Only feasible at specific points in the solution procedure, e.g. end of time step	Halt program at any time (almost) and dump
Solution procedure can often be modified before restarting	Continue the original job or nothing

Applications with built-in C/R
VASP

http://cms.mpi.univie.ac.at/vasp/vasp/ISTART_tag.html

- **ISTART-tag** ISTART=1 if WAVECAR exists, 0 otherwise
- This flag determines whether to read the file WAVECAR or not.
- 0 = Start job: begin from scratch. Initialize the orbitals according to the flag INIWAV.
- 1 = restart with constant energy cut-off. Continuation job - read orbitals from file WAVECAR
- 2 = restart with constant basis set : Continuation job -- read orbitals from the file WAVECAR
- 3 = full restart including orbitals and charge prediction
  Same as ISTART=2 but in addition a valid file TMPCAR must exist containing the positions and orbitals at time steps t(N-1) and t(N-2), which are needed for the orbital and charge prediction scheme (used for MD-runs).
- caveat – WAVECAR not always written frequently...

LAMMPS

http://lammps.sandia.gov/doc/restart.html

- **restart command**: checkpoint every so many timesteps

  restart 100000 restart.*.equil

  100000 => restart.1.equil

  200000 => restart.2.equil

  ...

  restart 1000 poly.1 poly.2

  1000 => poly.1

  2000 => poly.2

  3000 => poly.1

  4000 => poly.2

  ...

- **write_restart**: dump once
- **read_restart**: read and continue computation

GROMACS

http://www.gromacs.org/Documentation/How-tos/Doing_Restarts

- GROMACS writes restart files automatically (from v4.1 on): state.cpt
- Restart

```
mdrun -s topol.tpr -cpi state.cpt
```

Applications with built-in C/R

- Quantum-espresso
  - http://www.quantum-espresso.org/wp-content/uploads/Doc/pw_user_guide/node19.html
- ABINIT
  - http://www.abinit.org/doc/helpfiles/for-v7.10/input_variables/varrlx.html#restartxf
- CP2K
  - https://www.cp2k.org/restarting
- Molpro
  - https://www.molpro.net/info/2015.1/doc/quickstart/node65.html
- GAMESS
  - http://www.cfs.dl.ac.uk/docs/html/part3/node6.html
- OpenMx
  - http://www.openmx-square.org/adpack_man2.2/node18.html

- Check the application manual
  - Search for 'restart', rather than 'checkpoint'

Black box approaches

- Work "in principle" for any program
- In practice some caveats

1. BLCR (Berkeley lab C/R)
2. Intel mpirun (built-in BLCR)
3. dmtcp : distributed multi-threading checkpointing

- NO mpi
- intel mpi only
- ?

BLCR/csub

- Based on BLCR
- Use csub instead of qsub

```
qsub my_job_script.sh

csub --job_time="01:00:00" \
     --no_cleanup_chkpt    \
     -s my_job_script.sh

csub --resume="checkpoint_filename"
```

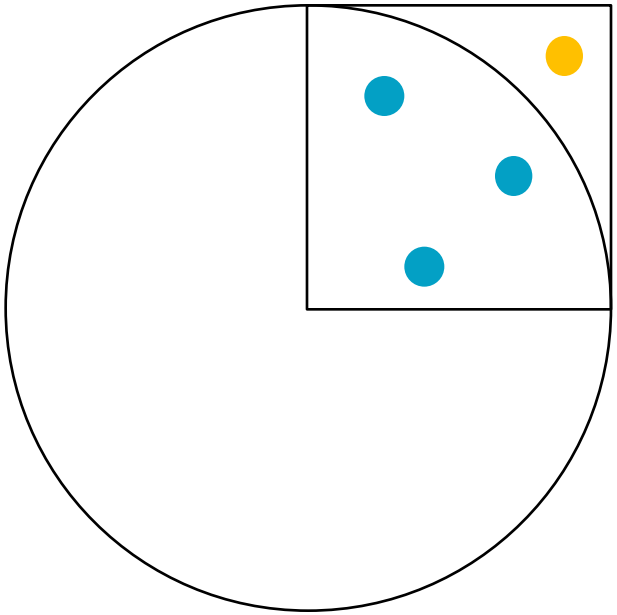
- Break up my_job_script.sh in 1h pieces and dump the program state after 1h
- Resume until job finishes
- Checkpoints are dumped in $VSC_SCRATCH/chkpt

Intel mpirun

- Allows to interrupt running program for checkpointing
- Only during
  - a mpi communication that involves ALL processes
  - MPI_COMM_WORLD
  - You can add MPI_BARRIER or MPI_IBARRIER+MPI_WAIT to your code to add more occasions for interrupting and checkpointing

Example problem

- Monte Carlo computation of π
- $A = \pi r^2$, $r = 1$
- Generate random point in square
- $\pi \sim 4\, n_{inside}/n_{total}$
- Easily parallellized
  - Make sure each thread/ or process has a random number generator with a different but deterministic seed



Intel mpirun

```
#!/bin/bash
#PBS -l nodes=2:ppn=20
#PBS -l walltime=0:10:00

cd $PBS_O_WORKDIR
mkdir -p ./ckpt # destination for checkpoint files

module load intel/2015a
export I_MPI_FABRICS=ofa        # some environment variables needed
export I_MPI_OFA_DYNAMIC_QPS=1
export I_MPI_OFA_NUM_RDMA_CONNECTIONS=0

mpirun -restart                              \# for restart only
       -ckpoint on                           \# checkpointing on
       -ckpoint-prefix $PBS_O_WORKDIR/ckpt \# checkpoint file dest
       -ckpoint-interval 60                  \# every 60 s
       ../../test                            \# executable
```

Case study

- VASP
- 4 nodes x 20 cores/node
- ~3h wall time
- $CuInSe_2$
- 4 atom system => 1 node per atom (rule of thumb)
- VASP/5.3.5-intel-2015a -> intel mpi

Job script

```
#!/bin/bash
#PBS -l nodes=4:ppn=20
#PBS -l walltime=01:00:00 #it will run out of walltime

module load VASP/5.3.5-intel-2015a
export I_MPI_FABRICS=ofa
export I_MPI_OFA_DYNAMIC_QPS=1
export I_MPI_OFA_NUM_RDMA_CONNECTIONS=0

cd $PBS_O_WORKDIR
mkdir -p ./ckpt

mpirun -verbose -ckpoint on
       -ckpoint-logfile ./ckpt.$PBS_JOBID.log
       -ckpoint-prefix  ./ckpt
       -ckpoint-interval 1200  #expect 2 chkpts
       vasp-eps2
```

Helps to diagnose errors

Files written

- vasp
  - ckpt
    - context-num1-0-0
    - context-num1-0-1
    - context-num1-0-2
    - context-num1-0-3
    - last-checkpoint-0-0
    - last-checkpoint-0-1
    - last-checkpoint-0-2
    - last-checkpoint-0-3
  - CHG
  - CHGCAR
  - ckpt.139912.mn.hopper.antwerpen.vsc.log
  - CONTCAR
  - DOSCAR
  - EIGENVAL
  - IBZKPT
  - INCAR
  - KPOINTS
  - OSZICAR
  - OUTCAR
  - pbs-vasp.sh
  - pbs-vasp.sh.e139912
  - PCDAT
  - POSCAR
  - POTCAR
  - vasprun.xml
  - WAVECAR
  - XDATCAR

Checkpoint files
1 for each node

Checkpoint log

could use VASP
built-in restart

Checkpoint log file

- vasp
  - ckpt
    - context-num1-0-0
    - context-num1-0-1
    - context-num1-0-2
    - context-num1-0-3
    - last-checkpoint-0-0
    - last-checkpoint-0-1
    - last-checkpoint-0-2
    - last-checkpoint-0-3
  - CHG
  - CHGCAR
  - ckpt.139912.mn.hopper.antwerpen.vsc.log
  - CONTCAR
  - DOSCAR
  - EIGENVAL
  - IBZKPT
  - INCAR
  - KPOINTS
  - OSZICAR
  - OUTCAR
  - pbs-vasp.sh
  - pbs-vasp.sh.e139912
  - PCDAT
  - POSCAR
  - POTCAR
  - vasprun.xml
  - WAVECAR
  - XDATCAR

[Thu Dec 10 10:06:38 2015] r3c6cn04.hopper.antwerpen.vsc Checkpoint log intialized (master mpiexec pid 54279, 80 processes, 4 nodes, keeping last 1 checkpoint(s))

[Thu Dec 10 10:06:38 2015] r3c6cn04.hopper.antwerpen.vsc Permanent checkpoint storage: /scratch/antwerpen/201/vsc20170/checkpointing/intel-mpi/vasp/ckpt

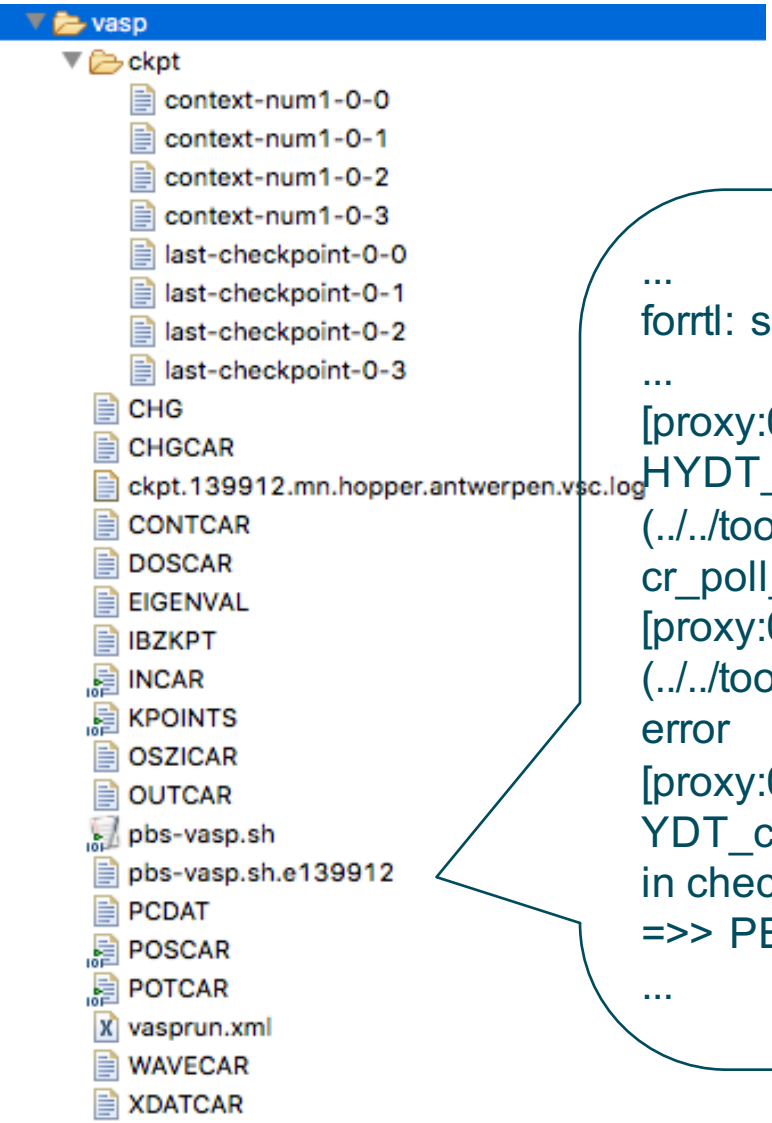
[Thu Dec 10 10:26:38 2015] r3c6cn04.hopper.antwerpen.vsc Started checkpoint number -1 ...

[Thu Dec 10 10:27:21 2015] r3c6cn04.hopper.antwerpen.vsc Finished checkpoint number -1.

[Thu Dec 10 10:47:21 2015] r3c6cn04.hopper.antwerpen.vsc Started checkpoint number 0 ...

Not finished?

Error file




```
...
forrtl: severe (174): SIGSEGV, segmentation fault occurred
...
[proxy:0:1@r4c3cn03.hopper.antwerpen.vsc]
HYDT_ckpoint_blcr_checkpoint
(../../tools/ckpoint/blcr/ckpoint_blcr.c:321):
cr_poll_checkpoint failed: Disk quota exceeded
[proxy:0:1@r4c3cn03.hopper.antwerpen.vsc] ckpoint_thread
(../../tools/ckpoint/ckpoint.c:620): blcr checkpoint returned
error
[proxy:0:1@r4c3cn03.hopper.antwerpen.vsc] H
YDT_ckpoint_finalize (../../tools/ckpoint/ckpoint.c:945): Error
in checkpoint thread 0x7
=>> PBS: job killed: walltime 3623 exceeded limit 3600
...
```

Let's restart…

```
#!/bin/bash
#PBS -l nodes=4:ppn=20
#PBS -l walltime=01:00:00 #it will run out of walltime
...
mpirun -verbose
       -restart
       -ckpoint on
       -ckpoint-logfile ./ckpt.$PBS_JOBID.log
       -ckpoint-prefix  ./ckpt
       -ckpoint-interval 1200
       vasp-eps2
```

- PBS: job killed: walltime 3635 exceeded limit 3600
- Restart again …

dmtcp

- Distributed multi-threading checkpointing
- Black box as intel mpirun, but program interrupts not limited to global MPI communication routines
- Promising – tests ongoing
- Not available as module yet, but installation is simple
- Requires template script
  - https://github.com/dmtcp
    - dmtcp/plugin/batch-queue/job_examples/torque_launch.job
    - dmtcp/plugin/batch-queue/job_examples/torque_rstr.job

dmtcp example

```
#!/bin/bash
# Put your PBS options here
#PBS -N dmtcp_example
#PBS -l nodes=2:ppn=2
...
# quite a bit of lines provided by dmtcp
...

start_coordinator --interval 60

restart=0 # or 1 if you want to restart
if [ $restart -eq 0 ]; then
   dmtcp_launch --rm --interval 60 ../t1-test1/test1
else
   ./dmtcp_restart_script.sh -h $DMTCP_COORD_HOST -p $DMTCP_COORD_PORT
fi
```

Checkpoint every 60s

Run: pbs-dmtpc-t1b.sh.o139941

```
===== start of prologue =====
Date : Thu Dec 10 13:15:55 CET 2015
Job ID : 139941
Job Name : pbs-dmtpc-t1b.sh
User ID : vsc20170
Group ID : vsc20170
Queue Name : q1h
Resource List : neednodes=1:ppn=1,nodes=1:ppn=1,walltime=00:05:00
===== end of prologue =======

PBS_JOBID=139941.mn.hopper.antwerpen.vsc
PBS_NODEFILE=/opt/moab/spool/torque/aux//139941.mn.hopper.antwerpen.vsc
r3c6cn04.hopper.antwerpen.vsc
PBS_O_WORKDIR=/scratch/antwerpen/201/vsc20170/checkpointing/dmtcp/t1b
which dmtcp_launch => /user/antwerpen/201/vsc20170/bin/dmtcp_launch

# processes                                : 1
m                                          : 10
std::numeric_limits<long  int   >::max()   : 9223372036854775807
std::numeric_limits<long  double>::max()   : 1.18973e+4932

===== start of epilogue =====
...
```

pbs-dmtpc-t1b.sh.e139941

```
[40000] NOTE at socketconnlist.cpp:178 in scanForPreExisting; REASON='found pre-existing
socket... will not be restored'
     fd = 11
     device = pipe:[1906658]
[40000] WARNING at socketconnection.cpp:192 in TcpConnection; REASON='JWARNING((domain ==
AF_INET || domain == AF_UNIX || domain == AF_INET6) && (type & 077) == SOCK_STREAM)
failed'
     domain = 0
     type = 0
     protocol = 0
[40000] NOTE at socketconnlist.cpp:178 in scanForPreExisting; REASON='found pre-existing
socket... will not be restored'
     fd = 16
     device = pipe:[1906660]
[40000] WARNING at socketconnection.cpp:192 in TcpConnection; REASON='JWARNING((domain ==
AF_INET || domain == AF_UNIX || domain == AF_INET6) && (type & 077) == SOCK_STREAM)
failed'
     domain = 0
     type = 0
     protocol = 0
=>> PBS: job killed: walltime 320 exceeded limit 300
```

Restart: pbs-dmtpc-t1b.sh.o139957

```
...
===== end of prologue =======

PBS_JOBID=139941.mn.hopper.antwerpen.vsc
PBS_NODEFILE=/opt/moab/spool/torque/aux//139941.mn.hopper.antwerpen.vsc
r3c6cn04.hopper.antwerpen.vsc
PBS_O_WORKDIR=/scratch/antwerpen/201/vsc20170/checkpointing/dmtcp/t1b
which dmtcp_launch => /user/antwerpen/201/vsc20170/bin/dmtcp_launch

# processes                                : 1
m                                          : 10
std::numeric_limits<long  int   >::max()   : 9223372036854775807
std::numeric_limits<long  double>::max()   : 1.18973e+4932
lhit0 : 7854002172

Number of Procs  used:      1
Number of Points used:      10000000000*1
hit:                                        7854002172
Estimate of Pi:                3.1416008688
   Error of Pi:                8.21521020676194e-06

===== start of epilogue =====
...
```

pbs-dmtpc-t1b.sh.e139957

```
[40000] NOTE at socketconnlist.cpp:178 in scanForPreExisting; REASON='found pre-existing
socket... will not be restored'
     fd = 11
     device = pipe:[1906658]
[40000] WARNING at socketconnection.cpp:192 in TcpConnection; REASON='JWARNING((domain ==
AF_INET || domain == AF_UNIX || domain == AF_INET6) && (type & 077) == SOCK_STREAM)
failed'
     domain = 0
     type = 0
     protocol = 0
[40000] NOTE at socketconnlist.cpp:178 in scanForPreExisting; REASON='found pre-existing
socket... will not be restored'
     fd = 16
     device = pipe:[1906660]
[40000] WARNING at socketconnection.cpp:192 in TcpConnection; REASON='JWARNING((domain ==
AF_INET || domain == AF_UNIX || domain == AF_INET6) && (type & 077) == SOCK_STREAM)
failed'
     domain = 0
     type = 0
     protocol = 0
```

Round up

Checkpoint to

- Recover from problematic situations (hopefully) without completely re-running (big) jobs
  - Save resources
  - Save power
  - Save fair share
  - Save time to solution

Best practices 1

- If your application has built-in restart capabilities, use them (always)
  - Advantages:
    - You can continue a job with different parameters
    - Checkpointing is cheap in terms of runtime and storage
  - Carefully read the manual, and check the application forum for possible issues
  - Perform some representative tests on restarting before you run a big job
  - If you have to restart, first make a backup of the checkpoint files
  - Ensure that you have sufficient quota to store at least two checkpoints
    - (or store them in a place where there are no quota, e.g. /tmp, but then you are responsible for picking up the checkpoints yourself)

Best practices 2

- If your application has no built-in restart capabilities
- Single node -> csub
- Multi-node -> intel mpirun or dmtcp
  - Use a version that is built with a recent intel toolchain

- Perform some representative tests on restarting before you run a big job
- If you have to restart, first make a backup of the checkpoint files
- Ensure that you have sufficient quota to store at least two checkpoints
  - Files are large : Gb/node
  - You run out of disk space

To be continued

- We hope to gain experience
  - Try it out and tell us about your successes and failures
- dmtcp tests ongoing