Amber20

Webpage

http://ambermd.org/

Version

Amber20 + AmberTools20-up3

Build Environment

- Intel Parallel Studio 2017 Update8 (MPI only)
- GCC 7.3.1 (devtoolset-7)
- CUDA 10.1.243

Files Required

- Amber20.tar.bz2
- AmberTools20.tar.bz2
- (AmberTools20 update.1-3; obtained while running update_amber script)

Build Procedure

```
#!/bin/sh

VERSION=20
TOOLSVERSION=20

INSTALL_DIR="/local/apl/lx/amber20-up0"
TARBALL_DIR="/home/users/${USER}/Software/AMBER/20"

PARALLEL=12

#-----------------------------------------------------------------------
module purge
module load intel_parallelstudio/2017update8
module load scl/devtoolset-7
module load cuda/10.1

export AMBERHOME=${INSTALL_DIR}
export CUDA_HOME="/local/apl/lx/cuda-10.1"

export LANG=C
export LC_ALL=C

# install directory has to be prepared before running this script
if [ ! -d $AMBERHOME ]; then
  echo "Create $AMBERHOME before running this script."
  exit 1
fi

# the install directory must be empty
if [ "$(ls -A $AMBERHOME)" ]; then
  echo "Target directory $AMBERHOME not empty"
  exit 2
fi

ulimit -s unlimited

# prep files
cd $AMBERHOME
bunzip2 -c ${TARBALL_DIR}/Amber${VERSION}.tar.bz2 | tar xf -
bunzip2 -c ${TARBALL_DIR}/AmberTools${TOOLSVERSION}.tar.bz2 | tar xf -
```

```
mv amber${VERSION}_src/* .
rmdir amber${VERSION}_src

# install python first. otherwise, update_amber failed to connect ambermd.org
./AmberTools/src/configure_python
AMBER_PYTHON=$AMBERHOME/bin/amber.python

# apply patches and update AmberTools
echo y | $AMBER_PYTHON ./update_amber --upgrade
$AMBER_PYTHON ./update_amber --update

echo "[GPU serial edition (two versions)]"
LANG=C ./configure --no-updates -cuda gnu
make -j${PARALLEL} install && make clean

echo "[GPU parallel edition (two versions)]"
LANG=C ./configure --no-updates -mpi -cuda gnu
make -j${PARALLEL} install && make clean
# GPU tests will be done elsewhere
# ccgpup cannot access external network, ccfep doesn't have GPGPUs

echo "[CPU serial edition]"
LANG=C ./configure --no-updates gnu
make -j${PARALLEL} install
. ${AMBERHOME}/amber.sh
make test.serial
make clean

echo "[CPU openmp edition]"
LANG=C ./configure --no-updates -openmp gnu
make -j${PARALLEL} install
make test.openmp
make clean

echo "[CPU parallel edition]"
LANG=C ./configure --no-updates -mpi gnu
make -j${PARALLEL} install
export DO_PARALLEL="mpirun -np 2"
make test.parallel
export DO_PARALLEL="mpirun -np 4"
cd test && make test.parallel.4proc

cd $AMBERHOME
make clean && chmod 700 src
```

Notes on Performance

GPU

- On P100 (jobtype=gpup), old AMBER18-bf16 (gcc4.8 + CUDA-9.1) is slightly faster than this new AMBER20-up0 (gcc7 + CUDA-10.1) by a few percents.
    - AMBER20-up0 built with CUDA-9.1 was apparently slower than the version built with CUDA-10.1.
    - GPU version performance was not affected by the GCC version as long as we know.
    - This slow performance is already reported in official mailing list.
- Contrary on V100 (jobtype=gpuv), new version AMBER20-up0 is faster than old one (AMBER16-bf16) by about 5 percents.
- FYI, Turing generation GPUs are reported to be suffering from the terrible performance degradation by ~15 % according to the official ML.

At this point, Amber20-up0 has some advantage over the old version only if V100 is used explicitly (jobtype=gpuv) or you want to use newly introduced function in Amber20.
Otherwise, Amber18-bf16 might be better. (Performance of Amber20 will be improved by the future pacth releases.)

CPU

- pmemd.MPI tested on DHF system
  - pmemd.MPI built with Intel compiler is apparently faster than that built with GCC. There are no significant differences in performance of intel17 and intel19 builds.
  - GCC versions 6-8 show similar performance. However, 4.8 build is slower than the others (versions 6-8 from scl).

Notes

- MPI+OpenMP CPU version of pmemd was skipped.
- "update_amber" failed when system python (python2) was used. We thus employed amber miniconda version of python upon running "update_amber".
  - system python3 works fine
  - python2 of anaconda2-2019Jul works fine
- GCC-8 + cuda version failed on GB tests. First step of GB energy (EGB) is wrong. Dynamics after the first step seems to be influenced.
  - GCC-6 and GCC-7 are free from this problem. (GCC 4.8.5 may also be free from this issue.)
- GPU version of GAMD failed. (both on serial and parallel; not always?)
  - Log lines such as GAMD = ********** appear. Or the value becomes strangely 0.0.
  - So far this does not depend on GCC version.
  - The reproducibility is not 100%? Other energies are not affected significantly. Therefore, the dynamics itself is not affected?
    - There might be syncing bug on CPU and GPU buffers? Or there might be problem upon reduction of the value? I dunno.
- CPU version with Intel compiler (tested on 17u8, 19u5) failed on test/dhfr bussi test (ntt=11; Bussi thermostat) with numerical error. Kinetic energy at 2nd step is apparently wrong.
  - This may imply thermostat function is completely miscompiled. We thus decided to avoid intel compiler.
- gcc7+mkl17 and intel17(+mkl)+gcc7 version failed on nab test with numerical error. I couldn't judge the importance of this error, though. (due to intel17 mkl?)
  - intel19(+mkl)+gcc8 version can pass nab test. However, intel19 version met strange "Program error"s in other tests. We thus avoided intel19.
  - gcc7 without mkl does not suffer from this issue.