Amber24 Update 3 + AmberTools 25

Webpage

http://ambermd.org/

Version

Amber 24 update 3, AmberTools 25

Build Environment

- GCC 13.3.1 (gcc-toolset-13)
- CUDA 12.8 update 1
- OpenMPI 4.1.8 (CUDA-aware)
- MKL 2025.1.0 (oneAPI 2025.1.0)
- (Gaussian 16 C.02; for QM/MM test)

Files Required

- Amber24.tar.bz2
- AmberTools25.tar.bz2

Build Procedure

```
#!/bin/sh

# Amber24 update 3 + AmberTools25
VERSION=24
UPDATE=u3
TOOLSVERSION=25
TOOLSUPDATE=

INSTALL_DIR="/apl/amber/${VERSION}${UPDATE}+at${TOOLSVERSION}${TOOLSUPDATE}"
WORKDIR="/gwork/users/${USER}/amber${VERSION}"
TARBALL_DIR="/home/users/${USER}/Software/AMBER/${VERSION}"
TOOLS_TARBALL_DIR="/home/users/${USER}/Software/AmberTools/${TOOLSVERSION}"

PARALLEL=12

#------------------------------------------------------------------------
module -s purge
module -s load gcc-toolset/13
module -s load openmpi/4.1.8/gcc13-cuda12.8u1
module -s load cuda/12.8u1
module -s load mkl/2025.1.0
module -s load gaussian/16c02

export LANG=C
export LC_ALL=C
ulimit -s unlimited

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

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

```
# prep files
cd $WORKDIR
if [ -d ambertools${TOOLSVERSION}_src ]; then
 mv -f ambertools${TOOLSVERSION}_src at_erase
 rm -rf at_erase &
fi

if [ -d amber${VERSION}_src ]; then
 mv -f amber${VERSION}_src amber_erase
 rm -rf amber_erase &
fi

tar xf ${TOOLS_TARBALL_DIR}/ambertools${TOOLSVERSION}.tar.bz2
tar xf ${TARBALL_DIR}/Amber${VERSION}.tar.bz2
mv amber${VERSION}_src/{examples,src,benchmarks} ambertools${TOOLSVERSION}_src
mv amber${VERSION}_src/test/{cuda,hip} ambertools${TOOLSVERSION}_src/test

# prep python and update
cd ambertools${TOOLSVERSION}_src

sed -i -e "1s/env python/env python3/" update_amber
./update_amber --update
yes | ./update_amber --upgrade
./update_amber --update

# typo...?
sed -i -e "s/a19 /a19,/" AmberTools/src/rism/amber_rism_interface.F90
sed -i -e "s/f7.0'/f7.0,'/" AmberTools/src/packmol_memgen/packmol_memgen/lib/ppm3/opm.f

# CPU serial with installation of tests
echo "[CPU serial edition]"
mkdir build_cpu_serial && cd build_cpu_serial
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DCUDA=FALSE \
  -DINSTALL_TESTS=TRUE \
  -DDOWNLOAD_MINICONDA=TRUE \
  -DBUILD_REAXFF_PUREMD=TRUE \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
#make -j${PARALLEL} install && make clean
make install && make clean
cd ../ && rm -rf build_cpu_serial

# mark its origin at installation directory
cd ${INSTALL_DIR}
ln -s ./miniconda ./miniforge

cd ${WORKDIR}/ambertools${TOOLSVERSION}_src

# reuse installed python
AMBER_PYTHON=${INSTALL_DIR}/bin/amber.python
eval "$(${INSTALL_DIR}/miniforge/bin/conda shell.bash hook)"

# CUDA, serial, gcc
echo "[GPU serial edition]"
mkdir build_gpu_serial && cd build_gpu_serial
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DCUDA=TRUE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
```

```
  -DUSE_CONDA_LIBS=TRUE \
  -DPYTHON_EXECUTABLE=${INSTALL_DIR}/miniconda/bin/python \
  -DCUDA_TOOLKIT_ROOT_DIR=${CUDA_HOME} \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
#make -j${PARALLEL} install && make clean
make install && make clean
cd ../ && rm -rf build_gpu_serial

# GPU parallel
echo "[GPU parallel edition]"
mkdir build_gpu_parallel && cd build_gpu_parallel
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=TRUE \
  -DCUDA=TRUE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DPYTHON_EXECUTABLE=${INSTALL_DIR}/miniconda/bin/python \
  -DCUDA_TOOLKIT_ROOT_DIR=${CUDA_HOME} \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
#make -j${PARALLEL} install && make clean
make install && make clean
cd ../ && rm -rf build_gpu_parallel

# CPU openmp
echo "[CPU openmp edition]"
mkdir build_cpu_openmp && cd build_cpu_openmp
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DOPENMP=TRUE \
  -DCUDA=FALSE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DPYTHON_EXECUTABLE=${INSTALL_DIR}/miniconda/bin/python \
  -DBUILD_REAXFF_PUREMD=TRUE \
  -DCHECK_UPDATES=FALSE
#make -j${PARALLEL} install && make clean
make install && make clean
cd ../ && rm -rf build_cpu_openmp

# CPU mpi (don't build mpi+openmp version)
echo "[CPU parallel edition]"
mkdir build_cpu_parallel && cd build_cpu_parallel
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=TRUE \
  -DOPENMP=FALSE \
  -DCUDA=FALSE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DPYTHON_EXECUTABLE=${INSTALL_DIR}/miniconda/bin/python \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
#make -j${PARALLEL} install && make clean
make install && make clean
cd ../ && rm -rf build_cpu_parallel
```

```
# ad hoc fix for #!python
cd ${INSTALL_DIR}
for f in bin/*; do
 grep -d skip -l "\#\!python" $f
 if [ $? == 0 ]; then
   sed -i -e "s|\#\!python|\#\!${INSTALL_DIR}/miniconda/bin/python|" $f
 fi
done

# append env variables setting
cd ${INSTALL_DIR}
cat <<EOF >> amber.sh

# rccs setting
CUDA_AMBER=/apl/cuda/12.8u1
export PATH=\$CUDA_AMBER/bin:\$PATH
export LD_LIBRARY_PATH=\$CUDA_AMBER/lib64:\$LD_LIBRARY_PATH

OMPI_AMBER=/apl/openmpi/4.1.8/gcc13-cuda12.8u1
export PATH=\$OMPI_AMBER/bin:\$PATH
export LD_LIBRARY_PATH=\$OMPI_AMBER/lib:\$LD_LIBRARY_PATH
export OMPI_MCA_btl=^openib

export LD_LIBRARY_PATH=/apl/oneapi/mkl/2025.1.0/lib:\$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/apl/oneapi/compiler-rt/2025.1.0/lib:\$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/apl/pbs/22.05.11/lib:\$LD_LIBRARY_PATH
EOF

cat <<EOF >> amber.csh

# rccs setting
set CUDA_AMBER=/apl/cuda/12.8u1
setenv PATH "\$CUDA_AMBER/bin:\$PATH"
setenv LD_LIBRARY_PATH "\$CUDA_AMBER/lib64:\$LD_LIBRARY_PATH"

set OMPI_AMBER=/apl/openmpi/4.1.8/gcc13-cuda12.8u1
setenv PATH "\$OMPI_AMBER/bin:\$PATH"
setenv LD_LIBRARY_PATH "\$OMPI_AMBER/lib:\$LD_LIBRARY_PATH"
setenv OMPI_MCA_btl ^openib

setenv LD_LIBRARY_PATH "/apl/oneapi/mkl/2025.1.0/lib:\$LD_LIBRARY_PATH"
setenv LD_LIBRARY_PATH "/apl/oneapi/compiler-rt/2025.1.0/lib:\$LD_LIBRARY_PATH"
setenv LD_LIBRARY_PATH "/apl/pbs/22.05.11/lib:\$LD_LIBRARY_PATH"
EOF

# run tests
cd ${INSTALL_DIR}
. ${INSTALL_DIR}/amber.sh
# now, $AMBERHOME should be $INSTALL_DIR

# parallel tests first
export DO_PARALLEL="mpirun -np 2"

make test.parallel && make clean.test

export DO_PARALLEL="mpirun -np 4"
cd test; make test.parallel.4proc; make clean; cd ../

unset DO_PARALLEL

# openmp tests
make test.openmp && make clean.test

# serial tests
make test.serial && make clean.test
```

```
## gpu tests
#export DO_PARALLEL="mpirun -np 2"
#make test.cuda.parallel; make clean.test # DPFP
#cd test; make test.cuda.parallel.SPFP; make clean; cd ../
#
#unset DO_PARALLEL
#make test.cuda.serial; make clean.test # DPFP
#cd test; make test.cuda.serial.SPFP; make clean; cd../
```

Test of GPU version was performed on ccgpu (equipped with A30*2)

```
#!/bin/sh

# AmberTools25
VERSION=24
UPDATE=u3
TOOLSVERSION=25
TOOLSUPDATE=
INSTALL_DIR="/apl/amber/${VERSION}${UPDATE}+at${TOOLSVERSION}${TOOLSUPDATE}"

#------------------------------------------------------------------------
module -s purge
module -s load gcc-toolset/13
module -s load openmpi/4.1.8/gcc13-cuda12.8u1
module -s load cuda/12.8u1
module -s load mkl/2025.1.0
module -s load gaussian/16c02

export LANG=C
export LC_ALL=C
ulimit -s unlimited

cd ${INSTALL_DIR}
. ${INSTALL_DIR}/amber.sh

# gpu tests
export DO_PARALLEL="mpirun -np 2"
make test.cuda.parallel; make clean.test # DPFP
cd test; make test.cuda.parallel.SPFP; make clean; cd ../

unset DO_PARALLEL
make test.cuda.serial; make clean.test # DPFP
cd test; make test.cuda.serial.SPFP; make clean; cd ../
```

Test results

Test results are available in /apl/amber/24u3+at25/logs.

- pbsa_cuda_cg: Segmentation fault - invalid memory reference.
  - This error is probably due to CUDA 12. Same error as previous version.
  - The calculation itself seems to be completed.
- gbsa_tfe (gbsa1_tfe, gbsa3_tfe), middle-scheme/4096wat of test_amber_*
  - These functions may not exist in PMEMD.
- rism and packmol_memgen tests abort if amber_rism_interface.f90 and opm.f are not fixed.
  - (Fortran runtime error: Missing comma between descriptors)
  - This error also happens in conda package (dacase::ambertools-dac=25 downloaded on May 15, 2025).
- Others seem to be minor numerical or format error.

Notes

- Although pmemd is now an independent package, it still can be built in the conventional way.