

## AMBER 22 update 4

### Webpage

<http://ambermd.org/>

### Version

Amber22 update 4, AmberTools 23 update 4

### Build Environment

- GCC 10.3.1 (gcc-toolset-10)
- CUDA 12.0
- OpenMPI 4.1.4 (HPC-X 2.11)
- (Gaussian 16 C.02; used only for QM/MM test)

### Files Required

- Amber22.tar.bz2
- AmberTools22.tar.bz2
  - (will be upgraded to AmberTools23 in the procedure below)
  - (miniforge will be employed instead of miniconda)
- patch-cmake-python (use miniforge instead of miniconda)

```
--- cmake/UseMiniconda.cmake.org 2022-05-27 09:43:57.000000000 +0900
+++ cmake/UseMiniconda.cmake 2022-05-27 09:56:28.000000000 +0900
@@ -84,11 +84,14 @@
     endif()
   endif()

-   set(MINICONDA_INSTALLER_FILENAME "Miniconda${PYTHON_MAJOR_RELEASE}-${MINICONDA_VERSION}-${CONTINUUM_SYSTEM_NAME}-
${CONTINUUM_BITS}.${INSTALLER_SUFFIX}")
+   #set(MINICONDA_INSTALLER_FILENAME "Miniconda${PYTHON_MAJOR_RELEASE}-${MINICONDA_VERSION}-${CONTINUUM_SYSTEM_NAME}-
${CONTINUUM_BITS}.${INSTALLER_SUFFIX}")
+   ## mkamiya: assume x86_64 Linux...
+   set(MINICONDA_INSTALLER_FILENAME "Miniforge${PYTHON_MAJOR_RELEASE}-${MINICONDA_VERSION}-Linux-x86_64.sh")

# location to download the installer to
set(MINICONDA_INSTALLER ${MINICONDA_DOWNLOAD_DIR}/${MINICONDA_INSTALLER_FILENAME})
- set(INSTALLER_URL "http://repo.continuum.io/miniconda/${MINICONDA_INSTALLER_FILENAME}")
+ #set(INSTALLER_URL "http://repo.continuum.io/miniconda/${MINICONDA_INSTALLER_FILENAME}")
+ set(INSTALLER_URL "https://github.com/conda-
forge/miniforge/releases/download/${MINICONDA_VERSION}/${MINICONDA_INSTALLER_FILENAME}")

# If we've already downloaded the installer, use it.
if(EXISTS "${MINICONDA_INSTALLER}")
```

### Build Procedure

```
#!/bin/sh

VERSION=22
TOOLSVERSION=22
# ambertools will be upgraded to tools 23

MINIFORGE_VERSION="23.1.0-4" # ad hoc custom version

INSTALL_DIR="/apl/amber/22u4"
WORKDIR="/gwork/users/${USER}/amber22"
TARBALL_DIR="/home/users/${USER}/Software/AMBER/22"

PATCHX=${TARBALL_DIR}/patch-cmake-python
```

```

PARALLEL=12

#-----
module -s purge
module -s load gcc-toolset/10
module -s load openmpi/4.1.4-hpcx/gcc10
module -s load cuda/12.0
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

# build directory must be empty
if [ "$(ls -A $WORKDIR)" ]; then
    echo "Target directory $WORKDIR not empty"
    exit 2
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 $WORKDIR
if [ -d amber${VERSION}_src ]; then
    mv -f amber${VERSION}_src amber_erase
    rm -rf amber_erase &
fi

bunzip2 -c ${TARBALL_DIR}/Amber${VERSION}.tar.bz2 | tar xf -
bunzip2 -c ${TARBALL_DIR}/AmberTools${TOOLSVERSION}.tar.bz2 | tar xf -

# do update/upgrade of Amber/AmberTools
cd amber${VERSION}_src
export AMBERHOME=${WORKDIR}/amber${VERSION}_src

sed -i -e "1s/env python/env python3/" update_amber
sed -i -e '137s//, encoding="utf-8\')' updateutils/patch.py
sed -i -e "s/\.Vupgrade/'python3', '\.Vupgrade'" updateutils/upgrade.py
python3 ./update_amber --update
yes | python3 ./update_amber --upgrade
python3 ./update_amber --update

patch -p0 < $PATCHX
sed -i -e "s/latest/${MINIFORGE_VERSION}/" cmake/PythonInterpreterConfig.cmake

# 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 \
    -DFORCE_INTERNAL_LIBS="arpack" \

```

```

-DBUILD_QUICK=TRUE \
-DCHECK_UPDATES=FALSE

make -j${PARALLEL} install && make clean
cd ../ && rm -rf build_cpu_serial

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

cd ${WORKDIR}/amber${VERSION}_src

# CUDA, serial, gcc
echo "[GPU serial edition]"
mkdir build_gpu_serial && cd build_gpu_serial
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DCUDA=TRUE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DANACONDA_BIN=${INSTALL_DIR}/miniforge/bin \
  -DCUDA_TOOLKIT_ROOT_DIR=${CUDA_HOME} \
  -DFORCE_INTERNAL_LIBS="arpack" \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE

make -j${PARALLEL} 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=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=TRUE \
  -DCUDA=TRUE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DANACONDA_BIN=${INSTALL_DIR}/miniforge/bin \
  -DCUDA_TOOLKIT_ROOT_DIR=${CUDA_HOME} \
  -DFORCE_INTERNAL_LIBS="arpack" \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE

make -j${PARALLEL} 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=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DOPENMP=TRUE \
  -DCUDA=FALSE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DANACONDA_BIN=${INSTALL_DIR}/miniforge/bin \

```

```

-DFORCE_INTERNAL_LIBS="arpack" \
-DBUILD_REAXFF_PUREMD=TRUE \
-DBUILD_QUICK=TRUE \
-DCHECK_UPDATES=FALSE

make -j${PARALLEL} 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=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=TRUE \
  -DOPENMP=FALSE \
  -DCUDA=FALSE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DANACONDA_BIN=${INSTALL_DIR}/miniforge/bin \
  -DFORCE_INTERNAL_LIBS="arpack" \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE

make -j${PARALLEL} install && make clean
cd ../ && rm -rf build_cpu_parallel

# ad hoc fix for python path (not all the exec files are fixed)
cd ${INSTALL_DIR}/bin
for f in *.py packmol-memgen; do
  sed -i -e 2i"#!${INSTALL_DIR}/miniconda/bin/python" -e 1d $f
done

# 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; ./test_amber_cuda_parallel.sh SPFP; make clean; cd ../
#
#unset DO_PARALLEL
#make test.cuda.serial && make clean.test # DPFP
#cd test; ./test_amber_cuda_serial.sh SPFP; make clean && cd ../

```

Following job script was used to submit the tests, since GPUs are not available in login servers.

```
#!/bin/sh
#PBS -l select=1:ncpus=16:mpiprocs=16:ompthreads=1:ngpus=2
#PBS -l walltime=24:00:00

# amber22 + AmberTools22
INSTALL_DIR="/apl/amber/22u4"

#-----
module -s purge
module -s load gcc-toolset/10
module -s load openmpi/4.1.4-hpcx/gcc10
module -s load cuda/12.0
module -s load gaussian/16c02

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

# run tests
cd ${INSTALL_DIR}
. ${INSTALL_DIR}/amber.sh

make clean.test

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

## Test Results

Please check logfiles in /apl/amber/22u4/logs. There seems to be no critical issues.

- gb8\_trx: possible failure due to gbalphap etc. is negligible. This is caused by the changes in default values (according to the discussion in official ML).
- qmmm\_Quick/QMMM\_MD\_cEw tests aborted upon termination. Calculation result itself seems to be completed without problem.
  - We didn't do detailed investigation regarding this issue. MPI is not concerned with this issue. Compiler (type/version) or CUDA version might be involved in this issue.
  - Error from Run.diala-wat-qmmm-cew script (undefined variable output etc.) does not seem to be the cause of the abort.

## Notes

- The procedure itself is almost the same as [the previous one](#).