

AMBER 24 update 3

Webpage

<http://ambermd.org/>

Version

Amber24 update 3, AmberTools 24 update 8

Build Environment

- GCC 13.1.1 (gcc-toolset-13)
- CUDA 12.4 update 1
- OpenMPI 4.1.8 (CUDA-aware)
- MKL 2025.0.0.1 (oneAPI 2025.0.1)
- (Gaussian 16 C.02; only for QM/MM test)

Files Required

- Amber24.tar.bz2
- AmberTools24.tar.bz2

Build Procedure

```
#!/bin/sh

VERSION=24
TOOLSVERSION=24

# amber24 upadte 3 + AmberTools24 update 8
INSTALL_DIR="/apl/amber/24u3"
WORKDIR="/gwork/users/${USER}/amber24"
TARBALL_DIR="/home/users/${USER}/Software/AMBER/24"

PARALLEL=12

#-----
module -s purge
module -s load gcc-toolset/13
module -s load openmpi/4.1.8/gcc13-cuda12.4u1
module -s load cuda/12.4u1
module -s load mkl/2025.0.0.1
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
```

```

fi

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

tar xf ${TARBALL_DIR}/Amber${VERSION}.tar.bz2
tar xf ${TARBALL_DIR}/AmberTools${TOOLSVERSION}.tar.bz2

# prep python and update
cd amber${VERSION}_src
#export AMBERHOME=${WORKDIR}/amber${VERSION}_src

sed -i -e "1s/env python/env python3/" update_amber
python3.9 ./update_amber --update
yes | python3.9 ./update_amber --upgrade
python3.9 ./update_amber --update

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

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

cd ${WORKDIR}/amber${VERSION}_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 \
  -DBUILD_QUICK=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 interpreter path
# FixCondaShebang does not work for some files on this system...
# (not due to symbolic links)
ORIGPATH=${WORKDIR}/amber24_src/build_cpu_serial/CMakeFiles/miniconda/install
ORIGPATH2=/lustre${ORIGPATH} # this might be involved...
NEWPATH=${INSTALL_DIR}/miniconda

```

```

cd ${INSTALL_DIR}
for f in bin/* miniconda/bin/*; do
  grep -l gwork $f
  if [ $? == 0 ]; then
    sed -i -e "s|${ORIGPATH2}|${NEWPATH}|g" \
      -e "s|${ORIGPATH}|${NEWPATH}|g" $f
  fi
done

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

# rccs setting
export LD_LIBRARY_PATH=${INSTALL_DIR}/lib:${INSTALL_DIR}/lib64:${INSTALL_DIR}/miniconda/lib:${LD_LIBRARY_PATH}

CUDA_AMBER=/apl/cuda/12.4u1
export PATH=${CUDA_AMBER}/bin:${PATH}
export LD_LIBRARY_PATH=${CUDA_AMBER}/lib64:${LD_LIBRARY_PATH}

OMPI_AMBER=/apl/openmpi/4.1.8/gcc13-cuda12.4u1
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/2025.0.0.1/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH=/apl/oneapi/compiler-rt/2025.0.4/lib:${LD_LIBRARY_PATH}
EOF

cat <<EOF >> amber.csh

# rccs setting
setenv LD_LIBRARY_PATH "${INSTALL_DIR}/lib:${INSTALL_DIR}/lib64:${INSTALL_DIR}/miniconda/lib:${LD_LIBRARY_PATH}"

set CUDA_AMBER=/apl/cuda/12.4u1
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-cuda12.4u1
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/2025.0.0.1/lib:${LD_LIBRARY_PATH}"
setenv LD_LIBRARY_PATH "/apl/oneapi/compiler-rt/2025.0.4/lib:${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

```

Tests

GPU version tests were performed on ccgpu (equipped with A30*2).

```
#!/bin/sh

INSTALL_DIR="/apl/amber/24u3"

#-----
module -s purge
module -s load gcc-toolset/13
module -s load openmpi/4.1.8/gcc13-cuda12.4u1
module -s load cuda/12.4u1
module -s load mkl/2025.0.0.1
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 log files are available in /apl/amber/24u3/logs. The results are almost identical to [previous installation](#).

Notes

- Miniforge is officially used now. Therefore, patch file to replace miniconda with miniforge is no longer necessary.
- "update_amber" command failed with UnicodeDecodeError when python 3.6 (system default version) was employed. Python 3.9 (non-default version) was used only for update_amber.
- Parallel make (make -j) failed with missing file error of amber-modules/pmemd/gbl_constants_mod.mod. Serial make works fine.
- Libraries in miniforge are not correctly referenced? miniconda/lib was added to LD_LIBRARY_PATH in amber.*sh files. Otherwise, some of command do not work.
 - PYTHON_EXECUTABLE specification is necessary when miniconda is reused? ANACONDA_BIN no longer works.