AMBER 24 update 1

Webpage

http://ambermd.org/

Version

Amber24 update 1, AmberTools 24 update 2

Build Environment

- GCC 13.1.1 (gcc-toolset-13)
- CUDA 12.4 update 1
- OpenMPI 4.1.6 (CUDA-aware)
- MKL 2024.1
- (Gaussian 16 C.02; used for QM/MM tests)

Files Required

- Amber24.tar.bz2
- AmberTools24.tar.bz2
- patch-cmake-python : replace miniconda with miniforge

```
--- cmake/UseMiniconda.cmake.orig       2024-04-27 04:56:41.000000000 +0900
+++ cmake/UseMiniconda.cmake    2024-06-06 13:12:23.000000000 +0900
@@ -92,11 +92,11 @@
             endif()
     endif()

-    set(MINICONDA_INSTALLER_FILENAME "Miniconda${PYTHON_MAJOR_RELEASE}-${MINICONDA_VERSION}-${CONTINUUM_SYSTEM_NAME}-
${CONTINUUM_BITS}.${INSTALLER_SUFFIX}")
+     set(MINICONDA_INSTALLER_FILENAME "Miniforge${PYTHON_MAJOR_RELEASE}-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/${MINICONDA_INSTALLER_FILENAME}")
+     set(INSTALLER_URL "https://github.com/conda-
forge/miniforge/releases/${MINICONDA_VERSION}/download/${MINICONDA_INSTALLER_FILENAME}")

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

Build Procedure

```
#!/bin/sh

VERSION=24
TOOLSVERSION=24

MINIFORGE_VERSION="latest" # ad hoc custom version

# amber24 + AmberTools24
INSTALL_DIR="/apl/amber/24u1"
WORKDIR="/gwork/users/${USER}/amber24"
TARBALL_DIR="/home/users/${USER}/Software/AMBER/24"

PATCHX=${TARBALL_DIR}/patch-cmake-python
PARALLEL=12

#------------------------------------------------------------------------
module -s purge
module -s load gcc-toolset/13
module -s load openmpi/4.1.6/gcc13-cuda12.4u1
```

```
module -s load cuda/12.4u1
module -s load mkl/2024.1
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

# 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 ./update_amber --update
yes | python3 ./update_amber --upgrade # reserved; there are no upgrades now
python3 ./update_amber --update

patch -p0 < $PATCHX

# CPU serial with installation of tests
echo "[CPU serial edition]"
mkdir build_cpu_serial && cd build_cpu_serial
cmake .. \
  -DCMAKE_INSTALL_PREFIX=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DCUDA=FALSE \
  -DINSTALL_TESTS=TRUE \
  -DDOWNLOAD_MINICONDA=TRUE \
  -DMINICONDA_VERSION=${MINIFORGE_VERSION} \
  -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

# 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=${INSTALL_DIR} \
  -DCOMPILER=GNU \
  -DMPI=FALSE \
  -DCUDA=TRUE \
  -DINSTALL_TESTS=FALSE \
  -DDOWNLOAD_MINICONDA=FALSE \
  -DUSE_CONDA_LIBS=TRUE \
  -DANACONDA_BIN=${INSTALL_DIR}/miniforge/bin \
  -DCUDA_TOOLKIT_ROOT_DIR=${CUDA_HOME} \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
make -j${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=${CUDA_HOME} \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
make -j${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 \
  -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 \
  -DBUILD_QUICK=TRUE \
  -DCHECK_UPDATES=FALSE
make -j${PARALLEL} 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)
cd ${INSTALL_DIR}
for f in bin/* miniconda/bin/*; do
 if [ -f $f ]; then
  $(head -n 1 $f | grep -q -- "^#\!.*python")
  if [ $? == 0 ]; then
    sed -i -e 2i"#!${INSTALL_DIR}/miniconda/bin/python" -e 1d $f
  fi
 fi
done
# pdb2pqr has a different problem...
sed -i -e 3i"''''exec' ${INSTALL_DIR}/miniconda/bin/python '\$0' '\$@'" -e 2d bin/pdb2pqr

# run tests (not gpu ones)
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
```

After the installation, CUDA, Open MPI, and MKL settings were added to "amber.sh" and "amber.csh" files.

Tests

GPU tests have been performed with the following script (on ccgpu).

```
#!/bin/sh

INSTALL_DIR="/apl/amber/24u1"

#------------------------------------------------------------------------
module -s purge
module -s load gcc-toolset/13
module -s load openmpi/4.1.6/gcc13-cuda12.4u1
module -s load cuda/12.4u1
module -s load mkl/2024.1
```

```
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 logs can be found at /apl/amber/24u1/logs. (Note: log of 4 processes tests are not available there. All the tests have passed without problem.) For GPU tests, there are some non-negligible failures (e.g. pbsa_cuda_cg in test_at_cuda_serial). Other minor numerical issues are not listed here.

- pbsa_cuda_cg: Segmentation fault - invalid memory reference. (at_cuda serial)
  - The calculation (minimization) itself was completed. There would be something wrong at the termination of process?
  - The same error happened for gcc10+cuda12.0 binary.
- cuda/gb_ala3 (irest1_ntt2_igb1_ntc2): significant error on 11 or later steps? vrand=10 may be related to this issue? (SPFP, DPFP)
- cuda/4096wat (vrand): significant error on 6 or later steps? vrand=5 may be related to this issue? (SPFP, DPFP)
- cuda/4096wat_oct (pure_wat_oct_nvt_ntt2): significant error on 6 or later steps? This also has vrand=5. (SPFP, DPFP)
- cuda/RAMD: wrong comparison? There might be changes in output file format? (SPFP, DPFP)

Notes

- Tested only with gcc-13 + cuda-12.4u1. We didn't try other settings for this version.
  - (gcc10+cuda12.0 (without MPI) was briefly checked for pbsa_cuda_cg problem above. There were no notable differences.)
- Shebang fix for bin/ and miniconda/bin files with FixCondaShebang does not work well for some of files. We thus do it by ourselves.
  - Symbolic link in the working directory path is one of the cause of this glitches. There would be some additional issues?
  - bin/pdb2pqr is not a target of FixCondaShebang. We should fix it by ourselves. (Is there any reason why it shouldn't be a python script?)
- lio, mbx, plumed are not tested.
- There seems to be no disadvantage to CUDA-aware MPI. We thus employed that.
- For MVAPICH2GDR_GPU_DIRECT_COMM, it seems to require CUDA-aware MPI and GPU Direct RDMA for this purpose (according to the source code). We may be able to try this without MVAPICH2-GDR.
  - PMEMD was built with Open MPI 4.1.6 (CUDA-aware) + cuda 12.4u1. The related source code (src/pmemd/src/cuda/gpu.cpp) was manually modified.
  - Tested on JAC and STMV systems using two GPUs on separate nodes.
    - Changing the source code degraded the performance. Single GPU runs are faster than multi-node 2 GPUs runs regardless of the code modification.
    - See http://mug.mvapich.cse.ohio-state.edu/static/media/mug/presentations/23/MUG23WednesdaySamKhuvis.pdf.
    - Our result is somewhat inconsistent with the reference above. There might be something wrong with our system setting or understanding.
  - Adding "--mca btl_openib_want_cuda_gdr 1" or/and "--mca btl_openib_cuda_rdma_limit 100000" option to "mpirun" do not help. (The former one is already enabled in the default setting according to "ompi_info --all")
  - Adding "--mca btl_openib_have_driver_gdr 1" (default value is false) option does not help, either.
    - There may be something wrong in our system, since "btl_openib_have_driver_gdr" parameter is false in the output of "ompi_info --all".
  - We didn't try mvapich2-gdr. GB calculations are not tried.

- We decided not to enable this flag based on the results above.