AMBER 24 update 1

ウェブページ

http://ambermd.org/

バージョン

Amber24 update 1, AmberTools 24 update 2

ビルド環境

- 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; QM/MM テストにのみ使用)

ビルドに必要なファイル

- Amber24.tar.bz2
- AmberTools24.tar.bz2
- patch-cmake-python：miniforge を使うためのパッチ

```
--- 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}")
```

ビルド手順

```
#!/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
```

上記手順実行後、amber.sh と amber.csh の末尾に必要な CUDA, Open MPI, MKL のパスを手動で追加しています。

テスト

GPU テストは以下のスクリプトで実行(@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 ../
```

実際のログは /apl/amber/24u1/logs 以下にあります。CPU 版の 4並列テストについても全て成功(ログは自動で保存されなかったため logs 以下にありません)。GPU 版については pbsa_cuda_cg (test_at_cuda_serial)で一件異常終了するなどしている。そのほかについては軽微な数値エラー程度であるため割愛します。

- pbsa_cuda_cg: Segmentation fault - invalid memory reference. (at_cuda serial)
  - ただし、計算(minimization)は最後まで終了していると思われる。終了時に何か問題があるのかもしれない
  - gcc10+cuda12.0でも同じように segmentation fault が発生
- cuda/gb_ala3 (irest1_ntt2_igb1_ntc2): 11 ステップ以降で結果がずれる？vrand=10 と関連？(SPFP, DPFP)
- cuda/4096wat (vrand): 6 ステップ以降で結果が大きくずれている？vrand=5 と関連して何か非互換な点がある？(SPFP, DPFP)
- cuda/4096wat_oct (pure_wat_oct_nvt_ntt2): 6 ステップ以降で結果が大きくずれている？これも vrand=5 がある。(SPFP, DPFP)
- cuda/RAMD: 比較しているものがおかしいように見える (SPFP, DPFP)

メモ

- 今回は gcc-13 + cuda-12.4u1 でのみ検証。他のバージョンについては調査せず。
  - (上記 pbsa_cuda_cg のエラー検証で gcc10+cuda12.0 (MPI 無し)で少しだけ検証していますが、特筆すべき違いは確認できていません。)
- FixCondaShebang による bin/, miniconda/bin の shebang 置き換えはうまく動かないため、手動で無理矢理実行。
  - パスに symbolic link が入っているとうまく置き換えが動作しないケースあり。他のタイミングでも何か起こっているように見える。
  - bin/pdb2pqr は FixCondaShebang の対象外だと思われる。
- lio, mbx, plumed については検証せず。
- CUDA-aware MPI にして特にデメリットはなさそうであるため、今回は CUDA-aware MPI を使用。
- MVAPICH2GDR_GPU_DIRECT_COMM については、コードを確認する限り CUDA-aware MPI で GPU Direct RDMA が有効であれば mvapich2-gdr でなくても同等の性能向上を期待できると思われる。以下の条件で簡単に検証
  - Open MPI 4.1.6 (CUDA-aware) + cuda 12.4u1 で src/pmemd/src/cuda/gpu.cpp を手動修正してビルド
  - JAC と STMV で 2 GPU を使ったノード間並列の条件でテスト実行。
    - コード修正前の方が実行速度は上。また、コード修正の有無によらず、2 GPU (マルチノード並列)よりも 1 GPU の方が速度が出ている。
    - http://mug.mvapich.cse.ohio-state.edu/static/media/mug/presentations/23/MUG23WednesdaySamKhuvis.pdf の記述とは一致するか
    - コード修正で明確に速度が落ちているため、上記資料の結果とは矛盾しているようにも思える。手順に何か問題があり、期待される条件が成立していないのかもしれない。
  - 実行時に --mca btl_openib_want_cuda_gdr 1 と --mca btl_openib_cuda_rdma_limit 100000 をつけた場合つけない場合で検証したが特に変化は見られず。ompi_info --all 前者はデフォルトで true と表示されるため既に有効だったと思われる。
  - ompi_info --all で false となっている btl_openib_have_driver_gdr については --mca btl_openib_have_driver_gdr 1 を指定した場合も明確な速度向上は見られない。
    - 標準の状態で btl_openib_have_driver_gdr は false になっているため、何かシステム側に問題があるかもしれない。
  - 今回は mvapich2-gdr では検証せず。GB 系の計算についても検証していない。
  - 以上の結果を踏まえて今回は無効とした。