NAMD 3.0b6 - MPI

ウェブページ

http://www.ks.uiuc.edu/Research/namd/

バージョン

3.0b6

ビルド環境

- GCC 12.1.1 (gcc-toolset-12)
- Intel MKL 2024.0
- Open MPI 4.1.6

ビルドに必要なファイル

- NAMD_3.0b6_Source.tar.gz
  - tcl, tcl-threaded は http://www.ks.uiuc.edu/Research/namd/libraries より取得
  - fftw については MKL を利用

ビルド手順

```
#!/bin/sh

VERSION=3.0b6
CHARM_VERSION=7.0.0
WORKDIR=/gwork/users/${USER}
SOURCEDIR=/home/users/${USER}/Software/NAMD/${VERSION}
NAME=NAMD_${VERSION}_Source
TARBALL=${SOURCEDIR}/${NAME}.tar.gz

LIBURL=http://www.ks.uiuc.edu/Research/namd/libraries
#FFTW=fftw-linux-x86_64
#FFTW_URL=${LIBURL}/${FFTW}.tar.gz
TCL=tcl8.5.9-linux-x86_64
TCL_URL=${LIBURL}/${TCL}.tar.gz
TCL_THREADED=tcl8.5.9-linux-x86_64-threaded
TCL_THREADED_URL=${LIBURL}/${TCL_THREADED}.tar.gz

#TARBALL_FFTW=${SOURCEDIR}/${FFTW}.tar.gz
TARBALL_TCL=${SOURCEDIR}/${TCL}.tar.gz
TARBALL_TCL_THREADED=${SOURCEDIR}/${TCL_THREADED}.tar.gz

PARALLEL=12

#-----------------------------------------------------------------
umask 0022

export LANG=""
export LC_ALL=C

module -s purge
module -s load gcc-toolset/12
module -s load mkl/2024.0
module -s load openmpi/4.1.6/gcc12

cd ${WORKDIR}
if [ -d ${NAME} ]; then
  mv ${NAME} namd_erase
  rm -rf namd_erase &
fi
```

```
tar zxf ${TARBALL}
cd ${NAME}
tar xf charm-${CHARM_VERSION}.tar

cd charm-v${CHARM_VERSION}

export CC=gcc
export CXX=g++
export F90=gfortran
export F77=gfortran
export MPICXX=mpicxx
export MPICC=mpicc
export MPIF90=mpif90
export MPIF77=mpif90

./build charm++ mpi-linux-x86_64 \
      --no-build-shared --with-production -j${PARALLEL}
cd mpi-linux-x86_64/tests/charm++/megatest
make pgm
mpirun -np ${PARALLEL} ./pgm
cd ../../../../
cd ../

tar zxf ${TARBALL_TCL}
mv ${TCL} tcl
tar zxf ${TARBALL_TCL_THREADED}
mv ${TCL_THREADED} tcl-threaded

./config Linux-x86_64-g++ \
       --charm-arch mpi-linux-x86_64 \
       --with-mkl \
       --with-python
cd Linux-x86_64-g++

make -j${PARALLEL}
make release
# install contents of Linux-x86_64-g++/NAMD_3.0b6_Linux-x86_64-MPI.tar.gz into /apl/namd/3.0b6-mpi manually
```

メモ

- Open MPI 5.0.1 を使うと速度が明確に落ちてしまったため、Open MPI 4.1.6 を使用
- Intel oneAPI 2023 Compiler (Classic)よりも GCC12 の方が少し速度が出ているため GCC を採用
- シングルノードで実行する場合は、こちらの mpi 版でなく、multicore(smp) 版の方が少し速度が出るかもしれません。
- EPYC Milan の CPU では AVX512 による高速化は有効になりません。
    - 速度を求める場合はシングルノード CUDA 版の利用をご検討ください。