Amber

Korištenje

Na klasteru su instalirane CPU (klasični procesori) i GPU (grafički procesori) varijante programskog paketa Amber u serijskoj i paralelnoj izvedbi.

Dostupni tipovi i verzije:

Tip	Verzija	Modul
CPU	16	amber/16
CPU	20	amber/20
CPU	22	amber/22
GPU	16	amber/16-gpu
GPU	20	amber/20-gpu
GPU	22	amber/22-gpu

CPU Amber

Paralelne izvedbe Amber poslova:

amber-cpu-parallel.sge

```
#$ -N amber-cpu-parallel
#$ -q p28.q
#$ -pe *mpi 14
#$ -cwd

module load amber/16

mpirun -np $NSLOTS -genv MV2_ENABLE_AFFINITY 0 -machinefile $TMPDIR/machines pmemd.MPI -O -i MD.in -o MDp.out -
p ionic_gelator.prmtop -c HEAT.rst -r MDp.rst -x MDp.mdcrd -inf MD.info
```

amber-cpu-parallel.sge

```
#$ -N amber-cpu-parallel
#$ -q p28.q
#$ -pe *mpi 14
#$ -cwd

module load amber/16

mpirun -np $NSLOTS -genv MV2_ENABLE_AFFINITY 0 -machinefile $TMPDIR/machines MMPBSA.py.MPI -O -i mmpbsa.in -o
rezultat.dat -sp ionicbox_C5_BF4.prmtop -cp com.top -rp rec.top -lp lig.top -y MD.mdcrd
```

GPU Amber

Serijska izvedba Amber poslova:

amber-gpu-serial.sge

```
#$ -N amber-gpu-serial
#$ -pe gpu 1
#$ -cwd

module load amber/16-gpu

cuda-wrapper.sh pmemd.cuda -O -i MD.in -o MD_8.out -p ionic_gelator.prmtop -c MD_7.rst -r MD_8.rst -x MD_8.
mdcrd -inf MD_8.info
```

Važno

Aplikacija (`pmemd.cuda`) u serijskoj izvedbi se mora pozivati s **cuda-wrapper.sh**!

Paralelna izvedba Amber poslova:

amber-gpu-parallel.sge

```
#$ -N amber-gpu-parallel
#$ -pe gpusingle 2
#$ -cwd

module load amber/16-gpu

mvapich-wrapper.sh pmemd.cuda.MPI -O -i MD.in -o MD_1.out -p ionic.prmtop -c MD.rst -r MD_1.rst -x MD_1.mdcrd -
inf MD_1.info
```

Važno

Aplikacija (pmemd.cuda.MPI) u paralelnoj izvedbi se mora pozivati s **mvapich-wrapper.sh**!

Preporuka

Koristiti paralelnu okolinu **gpusingle** kako bi svi zatraženi grafički procesori bili dodijeljeni s istog fizičkog čvora jer širenje paralelnih GPU Amber poslova na više fizičkih čvorova **ne doprinosi** performansama.

Instalacija

Amber 16

Programski paket Amber je podijeljen u dva dijela:

- **AmberTools** - skup javno dostupnih programa koji se distribuiraju pod slobodnom licencom GPL
- **Amber** - simulacijski program `pmemd` koji nije javno objavljen i distribuira se pod ograničenom licencom

Priprema instalacije

Priprema obuhvaća:

1. otpakiranje arhive `Amber16.tar.bz2`
    - sadrži izvorni kod simulacijskog programa `pmemd`
2. otpakiranje arhive `AmberTools16.tar.bz2`
3. instalacija Parallel NetCDF biblioteke
    - biblioteka je preduvjet za paralelnu izvedbu Ambera

Otpakiranje izvornog koda:

```
# tar -xjf Amber16.tar.bz2
# tar -xjf AmberTools16.tar.bz2
```

Instalacija Parallel NetCDF biblioteke

```
# git clone https://github.com/Parallel-NetCDF/PnetCDF.git
# cd PnetCDF
PnetCDF# autoreconf -i
PnetCDF# ./configure --prefix=/apps/PnetCDF
PnetCDF# make -j8
PnetCDF# make install
```

CPU Amber

Instalacija *serijske* izvedbe Amber je obavljena kompajliranjem izvornog koda s kompajlerom Intel 2019:

```
amber16# source amber.sh
amber16# module load intel/2019
amber16# ./configure -noX11 --with-python /usr/bin/python --python-install local intel
amber16# make -j8
```

Instalacija *paralelne* izvedbe Amber je obavljena kompajliranjem izvornog koda s kompajlerom Intel 2019 i MPI implementacijom MVAPICH 2.2:

```
amber16# source amber.sh
amber16# module load mpi/mvapich2-intel-2.2-x86_64
amber16# ./configure -rism -mpi -noX11 --with-pnetcdf /apps/PnetCDF --with-python /usr/bin/python --python-
install local intel
amber16# make -j8
```

GPU Amber

Instalacija *serijske* izvedbe Amber je obavljena kompajliranjem izvornog koda s kompajlerom Intel 2017 te paralelnim frameworkom CUDA-9.0:

```
amber16# module load cuda/9-0
amber16# module load intel/2017
amber16# ./configure -cuda intel
```

Instalacija *paralelne* izvedbe Amber je obavljena kompajliranjem izvornog koda s kompajlerom Intel 2017, MPI implementacijom MVAPICH-2.2 te paralelnim frameworkom CUDA-9.0:

```
amber16# module load mpi/mvapich2-intel2017-cuda90-2.2-x86_64
amber16# ./configure -cuda -mpi --with-pnetcdf /apps/PnetCDF intel
```

Amber 20

Priprema je odrađena na isti način kao i u verziji 16 otpakiravanjem slobodno te komercijalno dostupnih paketa. Novost u Amber 20 u odnosu na 16 jest izgradnja izvršnog oblika paketa pomoću sustava `cmake`. Pritom je zadržan i klasični način `autoconf` tako da Amber 20 podržava dva načina za kompajliranje i linkanje u izvršni oblik.

GPU i CPU Amber 20 su na klaster instalirane sljedećom kombinacijom:

kompajler	build alat	varijanta
intel/2017	autoconf	CPU avx
intel/2017	cmake	CPU avx2
gcc/8	cmake	GPU

Napomena

Amber dolazi s nekolicinom Python pomoćnih alata kojima u varijanti izgradnje s `cmake`, shebang interpreter linija pokazuje na Python interpreter u lokalnom direktoriju gdje se inicijalno kompajliranje odvijalo. Ako će se izvršni oblik premještati u drugi direktorij npr. `/apps`, tada je potrebno ispraviti shebang interpreter liniju svih Python alata.

CPU Amber

Instalacija serijske izvedbe Amber 20 se vrši kompajliranjem izvornog koda s Intel 2017 kompajlerom:

```
# module load intel/2017
# export AMBERHOME=$HOME/amber20_src
# cd $HOME/amber20_src/
amber20# ./configure -noX11 --with-python /usr/bin/python --python-install local intel
amber20# make -j8
```

Instalacija paralelne izvedbe Amber 20 se vrši kompajliranjem izvornog koda s Intel 2017 kompajlerom i MVAPICH-2.2 MPI implementacijom:

```
amber20# module load mpi/mvapich2-intel2017-2.2-x86_64
amber20# ./configure -rism -mpi -noX11 --with-pnetcdf /apps/PnetCDF --with-python /usr/bin/python --python-
install local intel
amber20# make -j8
amber20# make install
```

Kopiranje izvršnog oblika u željenu putanju:

```
amber20# cp -R bin/ lib/ include/ dat/ logs/ /destination
```

GPU Amber

GPU varijanta Amber 20 je kompajlirana s GCC-8 i MVAPICH 2.2.

Priprema:

```
amber20# module load gcc/8
amber20# module load cuda/10-1
```

Editiranje datoteke `build/run_cmake`:

```
# Assume this is Linux:

cmake $AMBER_PREFIX/amber20_src \
-DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber20-gpu \
-DCOMPILER=GNU \
-DMPI=FALSE -DCUDA=TRUE -DINSTALL_TESTS=FALSE \
-DDOWNLOAD_MINICONDA=TRUE -DMINICONDA_USE_PY3=TRUE \
2>&1 | tee cmake.log
```

- postavljanje `-DMPI=FALSE` te `-DCUDA=TRUE`

Kompajliranje serijske GPU varijante:

```
amber20# build/run_cmake && make -j8 && make install
```

Dodatna priprema za paralelnu GPU varijantu:

```
amber20# module load mpi/mvapich2-2.2-x86_64
amber20# export NCCL_HOME=/apps/nvidia/nccl/nccl_2.3.5-2+cuda10.0_x86_64
```

`build/run_cmake`:

```
# Assume this is Linux:

cmake $AMBER_PREFIX/amber20_src \
-DCMAKE_INSTALL_PREFIX=$AMBER_PREFIX/amber20-gpu \
-DCOMPILER=GNU \
-DMPI=TRUE -DNCCL=TRUE -DCUDA=TRUE -DINSTALL_TESTS=FALSE \
-DDOWNLOAD_MINICONDA=TRUE -DMINICONDA_USE_PY3=TRUE \
2>&1 | tee cmake.log
```

- postavljanje `-DMPI=TRUE`, `-DCUDA=TRUE` te `-DNCCL=TRUE`

Kompajliranje paralelne GPU varijante:

```
amber20# build/run_cmake && make -j8 && make install
```

Kopiranje izvršnog oblika u željenu putanju:

```
amber20# cp -R amber20-gpu/* /destination/amber/20/gpu
```

Ispravljanje Python shebang interpretera:

```
find /destination/amber/20/gpu -type f -name '*.py' -exec sed -i 's/originalna\/putanja/destination\/amber\/20\
/gpu/' {} \;
```