サンプルジョブの実行方法

最終更新: 2024/4/23

はじめに

RCCS では共通領域に導入したアプリケーションについて、サンプルジョブスクリプトを用意しています。
RCCS で用意したものをそのまま使う場合はもちろん、自身のディレクトリに導入した別バージョンを使う際のテンプレートとしても利用できます。
以下では Gromacs と GAMESS を例にして、サンプルをどのように実行するのかを説明します。
なお、Gaussian については、専用の g16sub/g09sub を用いた方法をまずはご覧ください。

導入済みアプリケーションのリスト

いくつかの方法があります。

1. ウェブページ上の情報を参照する

パッケージプログラム一覧のページに掲載されている情報で確認できます。

2. module avail コマンドの出力を見る

以下のように出力されます。通常、ジョブとして投入するようなアプリについては以下で赤で示した /apl/modules/apl 内にあります。

```
[user@ccfep3 ~]$ module avail
---------------------------- /apl/modules/defaults ----------------------------
2022

----------------------------- /apl/modules/oneapi -----------------------------
compiler-rt/2022.0.2  intelmpi/2021.7.1     mkl/2022.0.2  tbb/2021.7.1
compiler-rt/2022.2.1  intelpython/2022.0.0  mkl/2022.2.1
...
(中略)
...
------------------------------ /apl/modules/apl ------------------------------
amber/20u13         gromacs/2021.4-CUDA     nwchem/6.8
amber/22u1          gromacs/2021.6          nwchem/7.0.2
cp2k/9.1            gromacs/2021.6-CUDA     openmolcas/v21.10
cp2k/9.1-impi       gromacs/2022.4          openmolcas/v22.10
crystal/17-1.0.2    gromacs/2022.4-CUDA     orca/4.2.1
gamess/2021R1       GRRM/14-g09             orca/5.0.3
gamess/2022R2       GRRM/17-g09             qe/6.8
gaussian/09e01      GRRM/17-g16             qe/6.8-gpu
gaussian/16b01      lammps/2021-Sep29       reactionplus/1.0
gaussian/16c01      lammps/2021-Sep29-CUDA  siesta/4.1.5-mpi
gaussian/16c02      lammps/2022-Jun23       siesta/4.1.5-omp
genesis/2.0.3       lammps/2022-Jun23-CUDA  turbomole/7.6-mpi
genesis/2.0.3-CUDA  namd/2.14               turbomole/7.6-serial
gromacs/2021.4      namd/2.14-CUDA          turbomole/7.6-smp

Key:
loaded  directory/  auto-loaded  default-version  modulepath
[user@ccfep3 ~]$
```

(module avail コマンドはキーボードの q を押すか、画面の下までスクロールさせると終了します。)

3. /apl を参照する

OS 非標準のアプリやライブラリについては、/apl に導入されています。そちらで直接確認することもできます。
/apl/(アプリケーション名)/(バージョンやリビジョン) のようなディレクトリ構成となっています。

```
[user@ccfep3 /apl]$ ls
```

```
amber  crystal  gaussian  hpc-x   mvapich  nwchem     orca  reactionplus
aocc   cuda     genesis   lammps  namd     oneapi     pbs   siesta
aocl   dirac    gromacs   modules nbo      openmolcas psi4  turbomole
cp2k   gamess   GRRM      molpro  nvhpc    openmpi    qe    vmd
[user@ccfep3 /apl]$ ls /apl/amber
20u13  22u1
[user@ccfep3 /apl]$ ls /apl/amber/20u13
amber.csh   build          configure  lib64     README    test
amber.sh    cmake          dat        logs      recipe_at  update_amber
AmberTools  CMakeLists.txt  doc       Makefile  samples   updateutils
benchmarks  cmake-packaging  include  miniconda  share    wget-log
bin         config.h       lib        miniforge  src
[user@ccfep3 /apl]$
```

サンプルファイルの場所

各アプリ向けのサンプルファイルは原則として /apl/(アプリ名)/(バージョン)/samples 以下に置いています。

例: gromacs 2021.6 のサンプルを /apl から探す

```
[user@ccfep3 /apl]$ ls /apl
amber  crystal  gaussian  hpc-x   mvapich  nwchem     orca  reactionplus
aocc   cuda     genesis   lammps  namd     oneapi     pbs   siesta
aocl   dirac    gromacs   modules nbo      openmolcas psi4  turbomole
cp2k   gamess   GRRM      molpro  nvhpc    openmpi    qe    vmd
[user@ccfep3 /apl]$ ls /apl/gromacs/
2021.4  2021.4-CUDA  2021.6  2021.6-CUDA  2022.4  2022.4-CUDA
[user@ccfep3 /apl]$ ls /apl/gromacs/2021.6
bin  include  lib64  samples  share
[user@ccfep3 /apl]$ ls /apl/gromacs/2021.6/samples/
conf.gro    sample-mpi.csh  sample-threadmpi.csh  topol.top
grompp.mdp  sample-mpi.sh   sample-threadmpi.sh
```

(-CUDA がついているものは GPU 版です。)

サンプルディレクトリ内のファイルについて

サンプルディレクトリ内でインプットそのものは原則一つです。
ただし、それを実行するためのジョブスクリプトについては複数ある場合があります。
例えば、シェルが違う場合、並列方式が違う場合、設定の読み込み方法が違う場合、GPU利用の有無などが挙げられます。

- 例1: sample.sh => /bin/sh を使うサンプル
- 例2: sample.csh => /bin/csh を使うサンプル
- 例3: sample-gpu.sh => /bin/sh を使い、GPU も利用するサンプル

必要に応じて中身を比較するなどしてください。

例: gamess 2022R2 の場合

実行スクリプトが３つあります(sample.csh, sample-module.sh, sample.sh)。それぞれシェルの種類や設定方法が違いますが、インプットは exam01.inp 一つです。

```
[user@ccfep4 ~]$ ls /apl/gamess/2022R2/samples/
exam01.inp  sample.csh  sample-module.sh  sample.sh
```

例: gromacs 2021.6 の場合

こちらもいくつか sample スクリプトがありますが、インプットファイルは 1 セットだけです。

```
[user@ccfep4 samples]$ ls
conf.gro    sample-mpi.csh  sample-threadmpi.csh  topol.top
grompp.mdp  sample-mpi.sh   sample-threadmpi.sh
```

- -mpi がついているものは Open MPI (HPC-X)を使った並列版です(マルチノード対応)
- -threadmpi がついているものは組み込みの thread MPI を使った並列版です(シングルノード用)

サンプルの実行: 一般的な手順

- まず、サンプルのファイルを自分の書き込める領域にコピーします。
- サンプルをコピーしたディレクトリで jsub sample.sh のように実行します。
- (大抵の場合は、sh ./sample.sh のような形でログインノードで実行可能にもなっています。)
    - ただし、CPU 数の指定が jsub 時と変わる場合があります
    - GPU を使うものについては ccfep 上では実行できません。ccfep から ccgpu にログインしてお試しください(ssh ccgpu コマンド)。
    - ccgpu では GPU を 2 つ導入してありますので、MPI 並列テストも可能です。

例1: gamess 2022R2 の場合

サンプルを ~/gamess2022R2_test で実行する場合です。

```
[user@ccfep4 ~]$ mkdir -p ~/gamess2022R2_test
[user@ccfep4 ~]$ cd ~/gamess2022R2_test
[user@ccfep4 gamess2022R2_test]$ cp /apl/gamess2022R2/samples/* .
[user@ccfep4 gamess2022R2_test]$ ls
exam01.inp  sample-module.sh  sample.csh  sample.sh
[user@ccfep4 gamess2022R2_test]$ jsub sample.sh
4008689.cccms1
```

実行中のジョブの状況は jobinfo -c コマンドで確認できます。

```
[user@ccfep4 gamess2022R2_test]$ jobinfo -c
--------------------------------------------------------------------------------
Queue   Job ID Name           Status CPUs User/Grp       Elaps Node/(Reason)
--------------------------------------------------------------------------------
H       4008689 sample.sh      Run       4  user/---         -- ccc001
--------------------------------------------------------------------------------
[user@ccfep4 gamess2022R2_test]$
```

混雑していなければ、しばらく待てばジョブが終了し、出力結果が得られます。

```
[user@ccfep4 gamess2022R2_test]$ ls ~/gamess2022R2_test
exam01.dat  exam01.log  sample-module.sh  sample.sh.e4008689
exam01.inp  sample.csh  sample.sh         sample.sh.o4008689
[user@ccfep4 gamess2022R2_test]$
```

参考: sample.sh の中身について(青文字は説明文で、本来のファイルに含まれない部分です)

```
#!/bin/sh
#PBS -l select=1:ncpus=4:mpiprocs=4:ompthreads=1   # <= 4 コアのジョブ(1 vnode)
#PBS -l walltime=24:00:00   # <= 制限時間は 24 時間

if [ ! -z "${PBS_O_WORKDIR}" ]; then
  cd ${PBS_O_WORKDIR}  # <= ジョブを投入したディレクトリへ移動する。(jsub での投入時に必須の操作)
  NCPUS=$(wc -l < ${PBS_NODEFILE})
else
  NCPUS=4  # <= ここの二行は jsub を介さないで実行する時向けの設定
  export OMP_NUM_THREADS=1
fi

module -s purge
module -s load intelmpi/2021.7.1      #  <= 実行に必要な環境設定。アプリによって大きく変わります。
module -s load compiler-rt/2022.2.1

# processes per node; equal to mpiprocs value
PPN=4  # <= PPN = process per node, 一番上で定義されている mpiprocs の値と同じにする。

VERSION=2022R2
```

```
RUNGMS=/apl/gamess/${VERSION}/rungms
INPUT=exam01.inp

${RUNGMS} ${INPUT:r} 00 $NCPUS $PPN >& ${INPUT%.*}.log # <= 実際に実行するところ
```

例2: gromacs 2021.6 の場合

サンプルを ~/gromacs2021.6_test で実行する場合です。

```
[user@ccfep4 ~]$ mkdir -p ~/gromacs2021.6_test
[user@ccfep4 ~]$ cd ~/gromacs2021.6_test
[user@ccfep4 gromacs2021.6_test]$ cp /apl/gromacs/2021.6/samples/* .
[user@ccfep4 gromacs2021.6_test]$ ls
conf.gro    sample-mpi.csh  sample-threadmpi.csh  topol.top
grompp.mdp  sample-mpi.sh   sample-threadmpi.sh
[user@ccfep4 gromacs2021.6_test]$ jsub sample-mpi.sh
4008695.ccpbs1
```

実行中のジョブの状況は jobinfo -c コマンドで確認できます。

```
[user@ccfep3 gromacs2021.6_test]$ jobinfo -c
--------------------------------------------------------------------------------
Queue   Job ID Name            Status CPUs User/Grp       Elaps Node/(Reason)
--------------------------------------------------------------------------------
H       4008695 sample-mpi.sh  Run       6  user/---          -- ccc001
--------------------------------------------------------------------------------
```

混雑していなければ、しばらく待てばジョブが終了し、出力結果が得られます。

```
[user@ccfep4 gromacs2021.6_test]$ ls ~/gromacs2021.6_test
conf.gro     md.log           sample-mpi.sh.e4008695  topol.top
confout.gro  mdout.mdp        sample-mpi.sh.o4008695  topol.tpr
ener.edr     mdrun.out        sample-threadmpi.csh
grompp.mdp   sample-mpi.csh   sample-threadmpi.sh
grompp.out   sample-mpi.sh    state.cpt
[user@ccfep4 gromacs2021.6_test]$
```

参考: sample-mpi.sh の中身(青文字は説明文で、本来のファイルに含まれない部分です)

```
#!/bin/sh
#PBS -l select=1:ncpus=6:mpiprocs=6:ompthreads=1 # <= 6 コア(6 MPI ジョブ)
#PBS -l walltime=00:30:00  # <= 制限時間 30 分

if [ ! -z "${PBS_O_WORKDIR}" ]; then
  cd "${PBS_O_WORKDIR}"
fi

# non-module version
. /apl/hpc-x/2.13.1/hpcx-rebuild-gcc11.sh   # <= Open MPI (HPC-X)設定の読み込み
hpcx_load
export LD_LIBRARY_PATH="/apl/pbs/22.05.11/lib:${LD_LIBRARY_PATH}"
. /apl/gromacs/2021.6/bin/GMXRC  # <= Gromacs の環境設定の読み込み

## module version
#module -s purge
#module -s load --auto gromacs/2021.6  # <= module を使う場合、上記設定は module に定義されています

##############################################################################

N_MPI=6
N_OMP=1

gmx grompp -f grompp.mdp >& grompp.out
mpirun -v -n ${N_MPI} gmx_mpi mdrun -ntomp ${N_OMP} -s topol >& mdrun.out
```

ジョブスクリプトの書き方に関する参考情報

こちらのページに select 文のサンプルや簡単な説明があります。