Sample Jobs

Last update: Aug 4, 2021

Introduction

We prepared not only applications but also sample input/jobscripts. Those samples can also be used as a template file for your jobs. In the following, we will show you procedures how to run those sample jobs using GAMESS and Gromacs. As for Gaussian, please check the special way using g16sub/g09sub first.

List of Installed Applications

There are several ways to know which applications are installed.

1. at web page

You can see the application list in this page.

2. module avail

"module avail" command will give you a (long) list of applications. Applications which should be submitted as jobs would be listed in "apl_ex" category, which are colored with red in the box below.

```
[user@ccfep4 ~]$ module avail

------------------------------ /local/apl/lx/modules/suite -------------------------------
intel_parallelstudio/2015update1        intel_parallelstudio/2020update2
intel_parallelstudio/2017update4        scl/devtoolset-3
(中略)

------------------------------ /local/apl/lx/modules/comp --------------------------------
cuda/10.1            intel/15.0.1        intel/19.0.1        pgi/16.5
cuda/11.1            intel/17.0.4        intel/19.0.5        pgi/17.5
cuda/7.5             intel/17.0.8        intel/19.1.2        pgi/18.1(default)
cuda/8.0             intel/18.0.2        julia/1.3.1         pgi/20.4
cuda/9.1(default)    intel/18.0.5(default) julia/1.5.3

------------------------------ /local/apl/lx/modules/apl ---------------------------------
mpi/intelmpi/2017.3.196   mpi/openmpi/2.1.3/intel19 mpi/openmpi/4.0.0/gnu8.3
mpi/intelmpi/2017.4.262   mpi/openmpi/3.1.0/gnu4.8  mpi/openmpi/4.0.0/intel15
(...skipped)

------------------------------ /local/apl/lx/modules/apl_ex ------------------------------
GRRM/11-g09                 gromacs/2018.7/gnu
GRRM/14-g09(default)        gromacs/2018.7/gnu-CUDA
GRRM/17-g09                 gromacs/2018.7/intel
GRRM/17-g16                 gromacs/2018.7/intel-CUDA
abinit/7.8.2                gromacs/2018.8/gnu
abinit/8.8.3(default)       gromacs/2018.8/gnu-CUDA
amber/16/bugfix10           gromacs/2018.8/intel
amber/16/bugfix15           gromacs/2018.8/intel-CUDA
amber/18/bugfix1            gromacs/2019.2/gnu
amber/18/bugfix11-volta     gromacs/2019.2/gnu-CUDA
(...skipped)
gromacs/2018.3/gnu-CUDA     turbomole/7.4-serial
gromacs/2018.3/intel        turbomole/7.4.1-MPI(default)
gromacs/2018.3/intel-CUDA   turbomole/7.4.1-SMP
gromacs/2018.6/gnu          turbomole/7.4.1-serial
gromacs/2018.6/gnu-CUDA     turbomole/7.5-MPI
gromacs/2018.6/intel        turbomole/7.5-SMP
gromacs/2018.6/intel-CUDA   turbomole/7.5-serial

---------------------------- /local/apl/lx/modules/apl_viewer -----------------------------
luscus/0.8.6 molden/5.7   nboview/2.0  vmd/1.9.3

---------------------------- /local/apl/lx/modules/apl_util -------------------------------
allinea/7.1          cmake/3.16.3
cmake/2.8.12.2(default) cmake/3.8.2

------------------------------ /local/apl/lx/modules/lib ---------------------------------
boost/1.53.0(default)     mkl/2017.0.4              mkl/2020.0.2
boost/1.59.0              mkl/2018.0.2              nccl/2.3.7-1+cuda9.1(default)
boost/1.70.0              mkl/2018.0.4(default)     spglib/1.11.1(default)
mkl/11.2.1                mkl/2019.0.1
mkl/2017.0.3              mkl/2019.0.5

------------------------------ /local/apl/lx/modules/misc --------------------------------
inteldev intellic pgilic
[user@ccfep4 ~]$
```

3. see /local/apl/lx directly

Non-OS standard applications and libraries are installed under /local/apl/lx. You can directly check the list there.

```
[user@ccfep4 ~]$ ls /local/apl/lx
GRRM@                  gromacs2016.5-gnu/              nbo6018mod/
GRRM11/                gromacs2016.5-gnu-CUDA@         nbo70@
GRRM14/                gromacs2016.5-gnu-CUDA8/        nbo702/
GRRM17/                gromacs2016.6/                  nbo702-i4/
abinit@                gromacs2016.6-CUDA/             nbo707/
abinit782/             gromacs2016.6-gnu/              nbo707-i4/
abinit883/             gromacs2016.6-gnu-CUDA/         nbopro7/
(...skipped)
gromacs2016.4/         namd213/                        wine30/
gromacs2016.4-CUDA/    namd213-CUDA/                   wine30-win64/
gromacs2016.5/         nbo60@                          wine40-win64/
gromacs2016.5-CUDA@    nbo6015/
gromacs2016.5-CUDA9/   nbo6018/
[user@ccfep4 ~]$
```

(Some of them are symbolic links.)

Location of Sample Input

Location of sample input for a certain software can be known by using "module help (package name)". Or, directly go to samples/ directory under /local/apl/lx/(pckage name), where package name is usually (software name)(version).

Example 1: search for GAMESS 2019Sep30 sample using "module help".

```
[user@ccfep4 ~]$ module help gamess/2019Sep30
(...skipped...)
  Doc(local): /local/apl/lx/gamess2019Sep30/INPUT.DOC

SAMPLES:
  /local/apl/lx/gamess2019Sep30/samples

INFO:
(...skipped...)
[user@ccfep4 ~]$ cd /local/apl/lx/gamess2019Sep30/samples
[user@ccfep4 samples]$ ls
exam01.inp  sample.csh
[user@ccfep4 samples]$
```

Example 2: search gromacs 2019.6 directory in /local/apl/lx, and then go to its sample directory.

```
[user@ccfep4 ~]$ cd /local/apl/lx
[user@ccfep4 lx]$ ls -d gromacs*
gromacs                gromacs2018.3-CUDA      gromacs2019.4-gnu-CUDA
gromacs2016            gromacs2018.3-gnu       gromacs2019.6
gromacs2016.1          gromacs2018.3-gnu-CUDA  gromacs2019.6-CUDA
gromacs2016.1-CUDA     gromacs2018.6           gromacs2019.6-gnu
gromacs2016.3          gromacs2018.6-CUDA      gromacs2019.6-gnu-CUDA
gromacs2016.3-CUDA     gromacs2018.6-gnu       gromacs2020.2
(...skipped)
[user@ccfep4 lx]$ cd gromacs2019.6/samples
[user@ccfep4 samples]$ ls
conf.gro              sample-mpi-module.sh  sample-threadmpi-module.csh  sample-threadmpi.sh
grompp.mdp            sample-mpi.csh        sample-threadmpi-module.sh   topol.top
sample-mpi-module.csh  sample-mpi.sh        sample-threadmpi.csh
[user@ccfep4 samples]$
```

(package name with "-CUDA" is GPU-enabled version. "-gnu" ones are built with GCC (others are build with intel compiler).)

Files in Sample Directory

In a sample directory, there is only one input data set in principle. However, there can be several job scripts in a sample directory (same input but using different shell, hardware, setting method).

Examples:

- sample.sh => /bin/sh sample script
- sample.csh => /bin/csh sample script
- sample-gpu.sh => /bin/sh sample script using GPU

Reading and comparing those files might be helpful to you.

Example: gamess 2019Sep30

There is only one script (as shown above) for this application.

```
[user@ccfep4 samples]$ ls
exam01.inp  sample.csh
```

Example: gromacs 2019.6

There is only single input data set, but several job scripts are available.

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

- -mpi => parallel version with Intel MPI (multinode parallel possible).
- -threadmpi => thread MPI parallel version (multinode parallel not available).
- -module => use "module" command for environmental settings.

Run Sample: Basics

- copy files in sample directory to your directory.
- cd to the directory, where the copied files exist
- submit a job (e.g. jsub -q PN sample.sh)
- (In most of samples, you can run them on the frontend node (e.g. sh ./sample.sh))
  - # of CPUs might be different between "jsub" and "sh" cases.
  - GPU runs are not possible on frontend nodes (ccfep). (Please use ccgpup or ccgpuv.)

Example 1: gamess 2019Sep30

We here assume your sample directory is ~/gamess2019Sep30_test.

```
[user@ccfep4 ~]$ mkdir -p ~/gamess2019Sep30_test
[user@ccfep4 ~]$ cd ~/gamess2019Sep30_test
[user@ccfep4 gamess2019Sep30_test]$ cp /local/apl/lx/gamess2019Sep30/samples/* .
[user@ccfep4 gamess2019Sep30_test]$ ls
exam01.inp  sample.csh
[user@ccfep4 gamess2019Sep30_test]$ jsub -q PN sample.csh
4685953.cccms1
```

Status of running job can be checked with "jobinfo -c".

```
[user@ccfep4 gamess2019Sep30_test]$ jobinfo -c
--------------------------------------------------------------------------------
Queue   Job ID Name          Status CPUs User/Grp      Elaps Node/(Reason)
--------------------------------------------------------------------------------
PN      4685953 sample.csh    Run      4  ***/---         -- cccc123
--------------------------------------------------------------------------------
```

If the system is not terribly crowded, it will soon finish and you can get the result.

```
[user@ccfep4 gamess2019Sep30_test]$ ls ~/gamess2019Sep30_test
exam01.dat   exam01.log              sample.csh*         sample.csh.o4685953
exam01.inp   nodefile-4685953.cccms1  sample.csh.e4685953
[user@ccfep4 gamess2019Sep30_test]$
```

Reference: sample.csh (Explanation colored with blue is not involved in the original file.)

```
#!/bin/csh -f
#PBS -l select=1:ncpus=4:mpiprocs=4:ompthreads=1:jobtype=core # <= 4 cores on 1 node
#PBS -l walltime=24:00:00 # <= 30 minutes of time limit
#
#  Gamess is compiled with sockets and OpenMP enabled.
#
if ($?PBS_O_WORKDIR) then
cd ${PBS_O_WORKDIR} # <= you need to changedir to this when submitted via jsub
endif

set gamess = gamess2019Sep30
set RUNGMS = /local/apl/lx/${gamess}/rungms # <= setting of GAMESS
set INPUT = exam01.inp

if ($?PBS_O_WORKDIR) then
set nproc="nodefile-${PBS_JOBID}" # <= node list definition in case of jsub
uniq -c ${PBS_NODEFILE} | sed 's/^ *\([0-9]*\) *\(.*\)$/\2 \1/' > $nproc
else
set nproc=4 # <= in case jsub not used, specify number of cores here.
setenv OMP_NUM_THREADS 1
endif
${RUNGMS} ${INPUT:r} 00 $nproc >& ${INPUT:r}.log
```

This gamess is not build with MPI. But, to use nodes list provided by queuing system (PBS_NODEFILE), we employ MPI specification (mpiprocs=4).

Example 2: gromacs 2019.6

We here assume your sample directory is ~/gromacs2019.6_test.

```
[user@ccfep4 ~]$ mkdir -p ~/gromacs2019.6_test
[user@ccfep4 ~]$ cd ~/gromacs2019.6_test
[user@ccfep4 gromacs2019.6_test]$ cp /local/apl/lx/gromacs2019.6/samples/* .
[user@ccfep4 gromacs2019.6_test]$ ls
conf.gro               sample-mpi-module.sh  sample-threadmpi-module.csh  sample-threadmpi.sh
grompp.mdp             sample-mpi.csh        sample-threadmpi-module.sh   topol.top
sample-mpi-module.csh  sample-mpi.sh         sample-threadmpi.csh
[user@ccfep4 gromacs2019.6_test]$ jsub -q PN sample-mpi.sh
4684922.cccms1
```

Status of running job can be checked with "jobinfo -c".

```
[user@ccfep4 gromacs2019.6_test]$ jobinfo -c
--------------------------------------------------------------------------------
Queue   Job ID Name          Status CPUs User/Grp      Elaps Node/(Reason)
--------------------------------------------------------------------------------
PN      4684922 sample-mpi.sh  Run      6  ***/---         -- cccc123
--------------------------------------------------------------------------------
```

If the system is not terribly crowded, it will soon finish and you can get the result.

```
[user@ccfep4 gromacs2019.6_test]$ ls ~/gromacs2019.6_test
conf.gro     mdout.mdp                    sample-mpi.csh              state.cpt
confout.gro  mdrun.out                    sample-mpi.sh               topol.top
ener.edr     sample-mpi-module.csh        sample-threadmpi-module.csh  topol.tpr
grompp.mdp   sample-mpi-module.csh.e4684922  sample-threadmpi-module.sh  traj.trr
grompp.out   sample-mpi-module.csh.o4684922  sample-threadmpi.csh
md.log       sample-mpi-module.sh         sample-threadmpi.sh
[user@ccfep4 gromacs2019.6_test]$
```

Reference: sample-mpi.sh (Explanation colored with blue is not involved in the original file.)

```
#!/bin/sh
#PBS -l select=1:ncpus=6:mpiprocs=6:ompthreads=1:jobtype=core # <= 6 MPI * 1 OMP
#PBS -l walltime=00:30:00 # <= 30 minutes of time limit

if [ ! -z "${PBS_O_WORKDIR}" ]; then
  cd "${PBS_O_WORKDIR}" # <= cd to directory where jsub execed
fi

. /local/apl/lx/gromacs2019.6/bin/GMXRC # <= load gromacs env

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

N_MPI=6 # <= MPI process num; same value as mpiprocs in the header
N_OMP=1 # <= OpenMP thread num; same value as ompthreads in the header

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

Tips about job scripts

You can found some examples in https://ccportal.ims.ac.jp/en/node/2377.