Gaussian job submission using g16sub

National Institutes of Natural Sciences
Okazaki Research Facilities
Research Center for Computational Science (RCCS)

Changelog

- Jul 29, 2019 First version
- Jan 16, 2020 Update
- Feb 3, 2020 Add notes for %mem, %nprocshared, %cpu
- Feb 18, 2021 Update

Introduction

The aim of this document is to explain how to submit Gaussian jobs using “g16sub” RCCS command.

Table of Contents

Prerequisites

Following conditions must be satisfied beforehand.

- You can login to RCCS frontend node (ccfep).
- You can send your files to RCCS (via scp/sftp).
  - (Setting guide for above can also be found at quick start guide page.)
- Gaussian input file (.com, .gjf)
  - You don’t need to specify amount of memory (%MEM) or # of CPU cores (%CPU, %GPUCPU) if g09sub or g16sub is employed.

Sample Gaussian Input File

The following input file (ch3cl.gjf) is used in this sample.

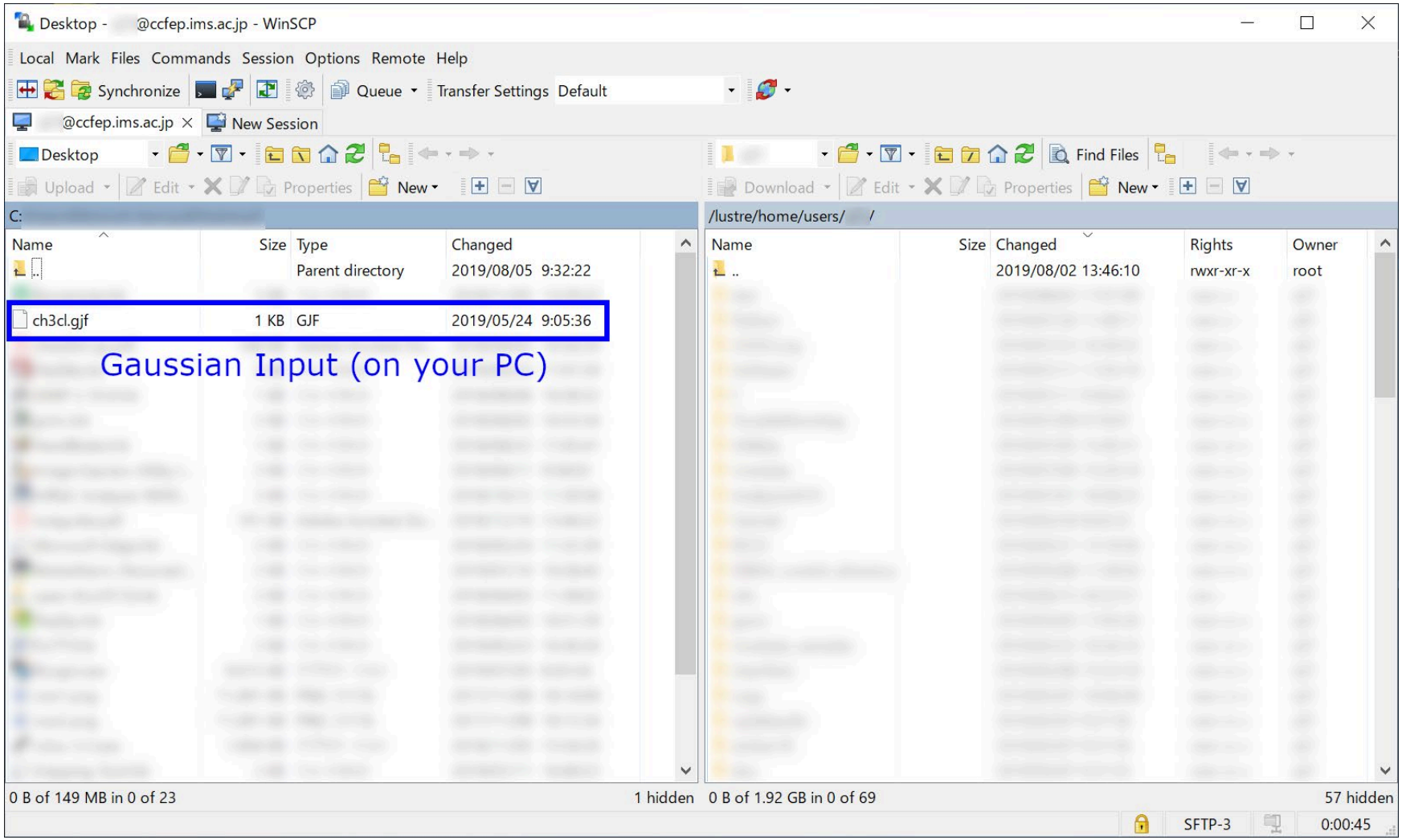
```
%chk=ch3cl.chk
# HF/6-31G(d,p) Opt

methyl chloride

0,1
C   -0.000004    1.127470    0.000000
H   -0.511417    1.468491    0.885898
H   -0.511417    1.468491   -0.885898
H    1.022922    1.468527    0.000000
Cl  -0.000004   -0.657078    0.000000
```

Usually, %mem, %nprocshared, %cpu settings are overwritten by g09sub/g16sub.
For CPU cores, please try -np (# of cores) option to specify the number.
For memory, you can skip it, since safe upper limit value is automatically chosen by g09sub/g16sub.

(Standard Gaussian input file extension is .gjf or .com.)

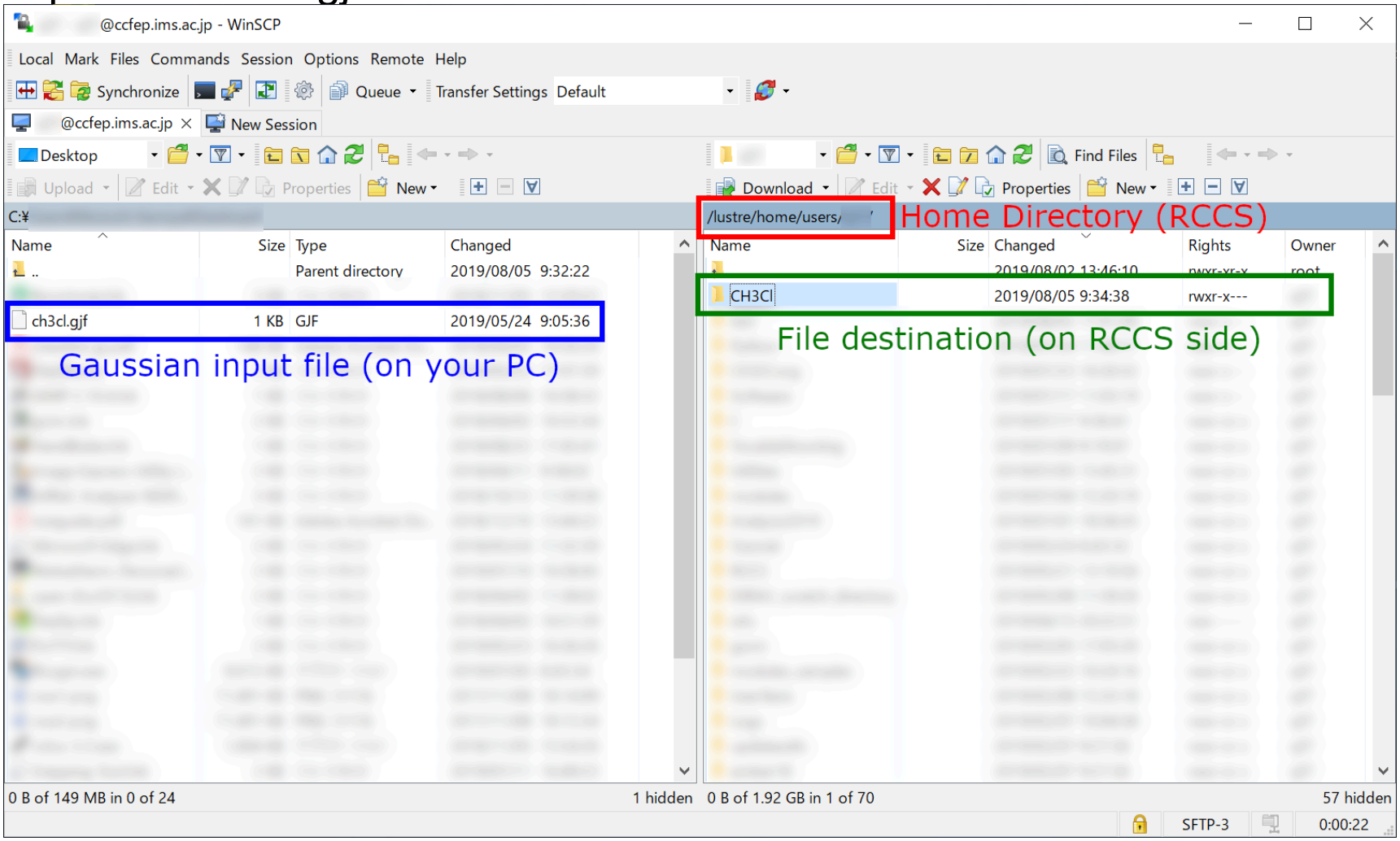
File Transfer (1)

In this document, we use WinSCP software to transfer file.



File Transfer (2)

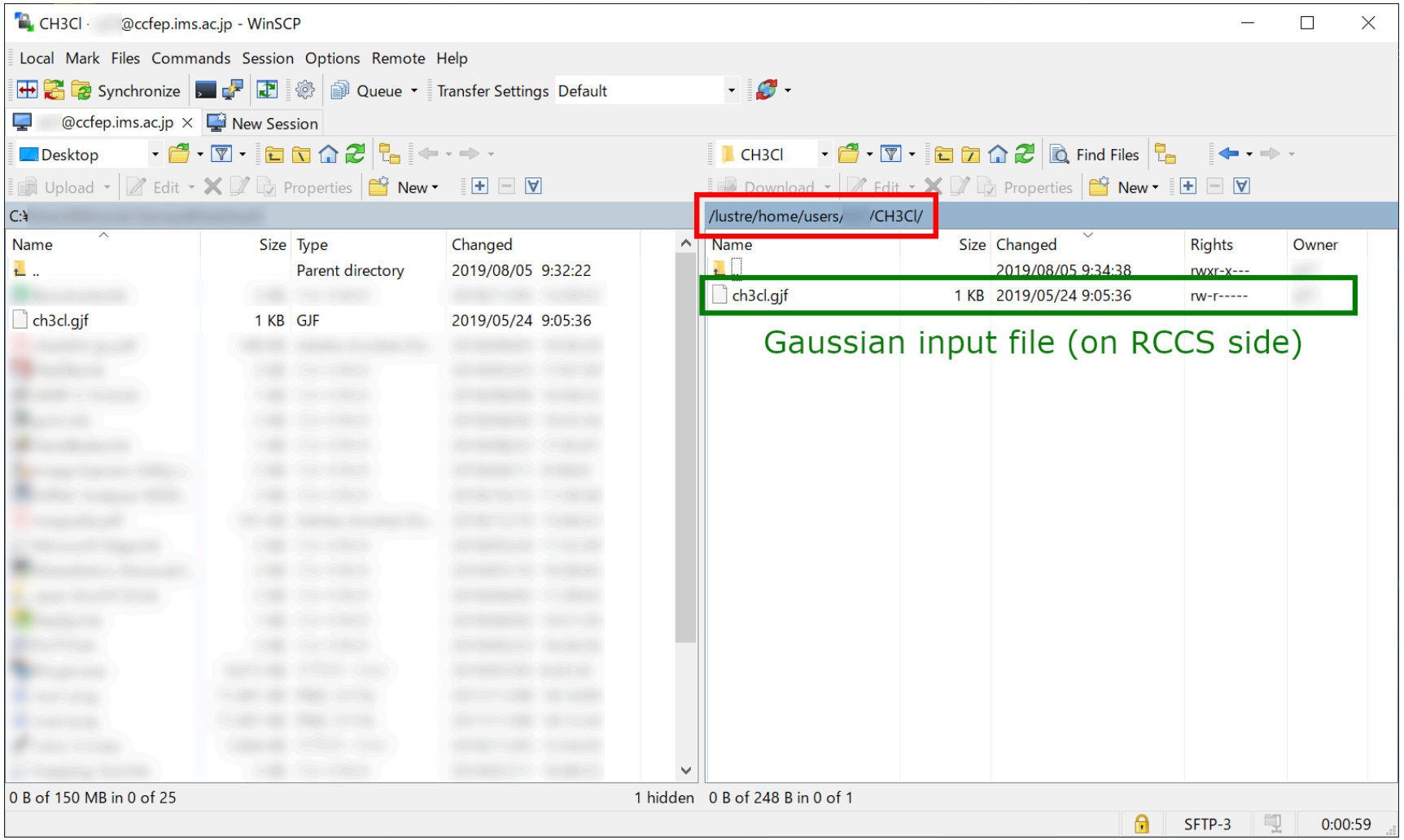
Make “CH3Cl” directory under your home directory at RCCS, and then put input file “ch3cl.gjf” there.



(“/home/users/(uid)” and “/lustre/home/users/(uid)” are the same place.)

File Transfer (3)

Once file transfer finished, quit WinSCP.
We will login via SSH and submit Gaussian job using g16sub.

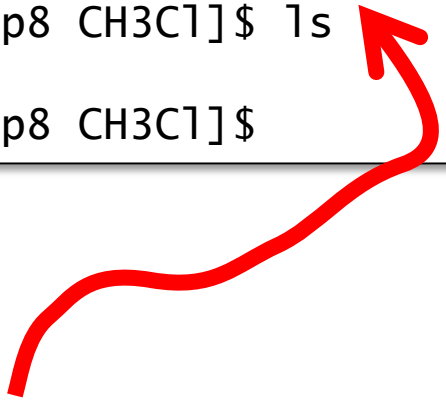


Login

(PuTTY is employed in this sample.)

Login to RCCS frontend (ccfep), and type “cd CH3Cl” and then Enter key to move to the directory where Gaussian input file resides.

```
Using username “***”.
Authenticating with public key “********************”
Passphrase for key “************************”:
Last login: Thu May 23 17:26:40 2019 from ******.****.**.**
[user@ccfep8 ~]$ ls CH3Cl/
ch3cl.gjf
[user@ccfep8 ~]$ cd CH3Cl/
[user@ccfep8 CH3Cl]$ ls
ch3cl.gjf
[user@ccfep8 CH3Cl]$
```



Move to the created directory using “cd” command.
File existence can be verified by “ls” command.

Submit Gaussian Job (1)

We can now submit a Gaussian job.
Submission command is “g16sub (inputfile)”.

```
Using username “***”.
Authenticating with public key “******************”
Passphrase for key “************************”:
Last login: Thu May 23 17:26:40 2019 from ******.****.**.**
[user@ccfep8 ~]$ ls CH3Cl/
ch3cl.gjf
[user@ccfep8 ~]$ cd CH3Cl/
[user@ccfep8 CH3Cl]$ ls
ch3cl.gjf
[user@ccfep8 CH3Cl]$ g16sub ch3cl.gjf
```

If you don’t add any other options, following default setting will be employed:

- jobtype=core
- 6 CPU cores
- 72 hours time limit

For further details of g16sub options, please check RCCS reference manual.

Reference Manual: https://ccportal.ims.ac.jp/en/QuickStart

Brief explanations about options can be found at the last part of this doc.

Submit Gaussian Job (2)

Once “g16sub” is submitted, various job information (# of cores, memory size, time limit) will be shown as follows (blue text area).

```
[user@ccfep8 CH3Cl]$ g16sub ch3cl.gjf
QUEUE detail
-------------------------------------------------------------------------------
QUEUE(MACH)  Jobtype  MaxMem     DefMem     TimLim     DefCPUs(Min-Max)
-------------------------------------------------------------------------------
   PN(  lx)  core     4.8GB      4.0GB      72:00:00   6(1-36)
-------------------------------------------------------------------------------
Job detail
===============================================================================
 MOL name(s)     : ch3cl
 INP file(s)     : ch3cl.gjf.lx
 OUT file(s)     : ch3cl.out
 Current dir     : /lustre/home/users/***/CH3Cl
 SCRATCH dir     : /work/users/${USER}/${PBS_JOBID}

 QUEUE           : PN
 Memory          : 24.0GB
 Time limit      : 72:00:00
 Job script      : /lustre/home/users/***/CH3Cl/PN_28524.sh
 Input modified  : y
===============================================================================

/usr/local/bin/jsub -q PN /lustre/home/users/***/CH3Cl/PN_28254.sh

4529602.cccms1
[user@ccfep8 CH3Cl]$
```

(queue params)

(job information)

Job ID

At the bottom, “(number).cccms1” will be shown, where (number) is called Job ID.
If there is any error on job submission, error message will be shown instead of Job ID.

Check Job Status

The status of your jobs can be checked with “jobinfo” command.
(On immediately after job submission, that job might not be shown. Please wait for a while.)
“jobinfo -q PN -c -l” is the standard usage.

```
[user@ccfep8 CH3Cl]$ jobinfo -q PN -c -l
-------------------------------------------------------------------------------
Queue    Job ID Name             Status CPUs User/Grp      Elaps Node/(Reason)
-------------------------------------------------------------------------------
PN     4529602 PN_28254.sh       Run ②     6  ***/---    00:00:00 cccc120 ④
---------①-------------------------------------------------③--------------------
[user@ccfep8 CH3Cl]$
```

Status of unfinished jobs will be shown along with their job IDs.

1. Unique job ID. This will be used when you stop job.
2. Job status. “Run”: running on computation node(s), “Queue”: waiting.
3. Duration time. If the status is “Run”, execution time up to now will be shown.
   If the status is “Queue”, total waiting time will be shown.
4. Name(s) of computation nodes will be shown when job is running.
   Otherwise, if job is waiting, reason of the wait will be shown.
   For example, (cpu) means job cannot be launched since not enough CPU cores are available. (Immediately after the submission, the reason may be (other).)

Job completed?

After the job completion, there remains some files as in the below sample. “PN_(number).sh” and “ch3cl.gjf.lx” are created by g16sub, and are less important if Gaussian calculation finished successfully.

```
[user@ccfep8 CH3Cl]$ ls
PN_28254.sh           PN_28254.sh.o4529602  ch3cl.gjf     ch3cl.out
PN_28254.sh.e4529602  ch3cl.chk             ch3cl.gjf.lx
[user@ccfep8 CH3Cl]$
```

Output file “ch3cl.out” can be checked even when the job is running. “less” and “tail” commands will be useful to check output.

If checkpoint file (.chk) name is specified in the input as in the example “ch3ch.gjf”, checkpoint file can also be found in this directory.

Run formchk

Checkpoint file (.chk) can be converted to formatted checkpoint file (.fchk) by using “formchk” command. In RCCS system, you need to load setting file beforehand. The preparation command depends on your shell type.

csh (/bin/csh):

```
[user@ccfep8 CH3Cl]$ source /local/apl/lx/g16/g16/bsd/g16.login
```

bash/zsh (/bin/bash or /bin/zsh):

```
[user@ccfep8 CH3Cl]$ source /local/apl/lx/g16/g16/bsd/g16.profile
```

Please be sure that the commands above don’t return any outputs. Prompt will be back promptly. If there are some output, it might be an error. Please check carefully.

Then, you can run formchk.

```
[user@ccfep8 CH3Cl]$ formchk ch3cl.chk ch3cl.fchk
 Read checkpoint file ch3cl.chk
 Write formatted file ch3cl.fchk
 FChkPn:  Coordinates translated and rotated
 FChkPn:  Coordintes match /B/ after translation and rotation
[user@ccfep8 CH3Cl]$
```

Tips (1): g16sub options

- g16sub default settings
  - “core” jobtype (-j core)
    - Run on single node. Your job can share resources with other user’s jobs.
  - 6 CPU cores (-np 6)
  - 72 hours time limit (-walltime 72:00:00)
  - Default Gaussian 16 revision is...
    - FY2019, Gaussian 16 Rev. B.01 (-rev g16b01)
    - FY2020-2021, Gaussian 16 Rev. C.01 (-rev g16c01)
      - (if -rev g16a03 specified, Gaussian 16 Rev. A.03 will be used.)
    - If you specify only Gaussian input file, that is equivalent to the following one.

    - FY2020-2021:
      g16sub -j core -rev g16c01 -np 6 -walltime 72:00:00 (input file name)

  - You can specify # of cpu cores and time limit using these options.

Tips (2): Other versions/revisions of Gaussian

- Following versions and revisions of Gaussian are available in RCCS. You can choose one using -rev option of g09sub/g16sub.
  - Gaussian 16 C.01 (g16sub -rev g16c01 ; default for g16sub)
  - Gaussian 16 B.01 (g16sub -rev g16b01)
  - Gaussian 16 A.03 (g16sub -rev g16a03)
  - Gaussian 09 E.01 (g09sub -rev g09e01 ; default for g09sub)
  - Gaussian 09 D.01 (g09sub -rev g09d01)
  - Gaussian 09 C.01 (g09sub -rev g09c01)
  - Gaussian 09 B.01 (g09sub -rev g09b01)

Tips (3): # of CPU cores

- (We basically assume jobtype = core in the following.)
  - Linda is not available in RCCS. Therefore, inter node parallel Gaussian runs are impossible even if jobtype = small or large employed.
- More of CPU cores does not mean more fast run. It can be even slower if too many CPU cores are employed.
- In terms of c cost-performance ratio, employing small number of CPU cores is usually more efficient.
- Moreover, in the aspect of job waiting time, smaller jobs are advantageous.
- However, job won't finish within the walltime limit if too small # of cores employed...
- "The best # of cores" can't be determined merely from the benchmark result. It often depends on your situation.
  - E.g.: you need the result urgently because deadline is coming. In this case, you might want to use many cores disregarding the cost.
  - E.g.: you can wait for three days because you have other tasks to do. In this case, you might want to reduce # of cores to improve cost efficiency.

Tips (4): Working Directory

Working directory is not yet determined before job submission.
SCRATCH dir name shown on g16sub output is not a complete information.

```
OUT file(s)     : ch3cl.out
Current dir     : /lustre/home/users/***/CH3Cl
SCRATCH dir     : /work/users/${USER}/${PBS_JOBID}
```

In the actual job, $USER and $PBS_JOBID will be replaced by your user ID (three-letter user name) and job ID (4529602.cccms1 for example), respectively.

ch3cl.out:

```
Entering Gaussian System, Link0=/local/apl/lx/g16b01/g16/._.g16
Initial command:
/local/apl/lx/g16b01/g16/l1.exe "/work/users/***/4529602.cccms1/Gau-33628.inp"
-scrdir="/work/users/***/4529602.cccms1/"
Default CPUs for threads: 6,7,8,9,10,11
Default is to use a totoal of  6 processors:
```

Actual scratch directory location is also confirmed in output file (ch3cl.out).

Please note that this scratch directory will be removed if Gaussian job submitted by g09sub/g16sub is finished within the specified "walltime".

Tips (5): Method other than g**sub

- You can submit Gaussian jobs without using g09sub/g16sub.
- Among those ways, using a template in /local/apl/lx/g16c01/samples might be the easiest (samples are available for revisions other than "g16c01").
  - You should modify the input file name and resource values.
  - # of CPU cores and GPUs should ALSO be specified in GAUSS_CDEF and GAUSS_GDEF environment variables, respectively.
  - Memory amount can be specified by -m command line option, or GAUSS_MDEF environment variable. (You can use %Mem link0 command if you add –noedit option for jsub.)
- If -P option is given in g16sub/g09sub, modified input file and job script will be generated but jobs is not submitted. You can use those files as a template for your run.
  - The generated jobscript (PN_*.sh) has a similar structure to that of sample located in /local/apl/lx/(Gaussian ver/rev)/samples.