g16subコマンドによるGaussianジョブの投入方法

最終更新: 2021/2/18

はじめに

このページでは g16sub コマンドで RCCS スパコンに Gaussian ジョブを投入する方法を説明します。

事前に必要な準備

前提条件として、以下の準備が必須です。

- RCCS スパコンフロントエンド(ccfep)にログインできるようにする
- scp, sftp 等で RCCS とのファイル送信をできるようにしておく
- Gaussian のインプットファイルを用意する(テスト用途ならば以下のサンプルインプットでも大丈夫です)

ssh や scp, sftp の設定ができていない場合は クイックスタートガイドのページをご覧ください。

サンプル Gaussian インプットファイル

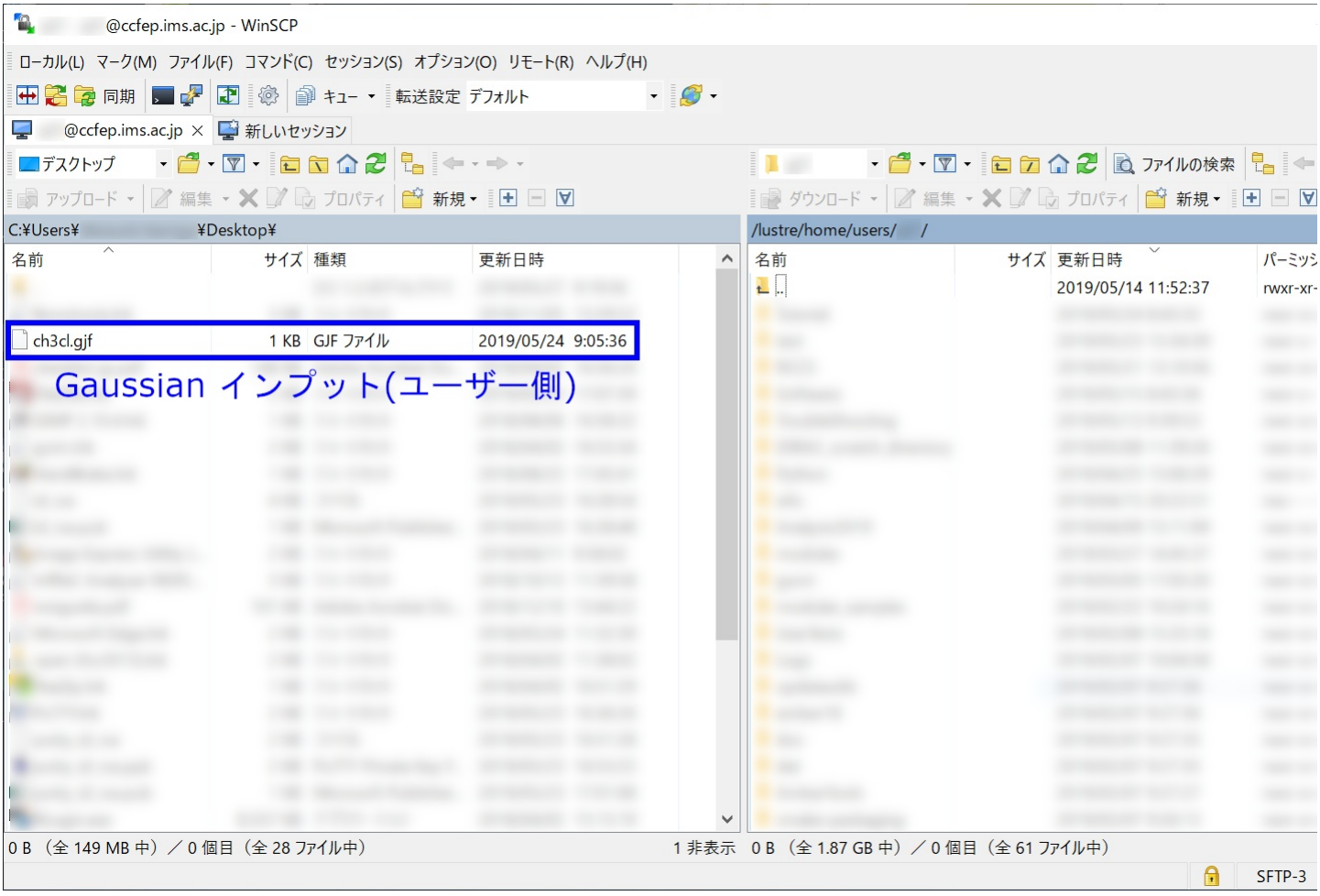
以下のような内容の ch3cl.gjf (.gjf は .com と同じく Gaussian インプットの標準的な拡張子です)が手元にあるとして、これを RCCS に転送し、Gaussian で計算を行います。

```
%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   0.0000
```

%mem, %nprocshared, %cpu の指定は通常 g09sub や g16sub に上書きされます。利用する CPU コア数については g16sub/g09sub の -np オプションで指定してください。メモリについては、jobtype 等に応じた上限値(少しだけ余裕をもたせてあります)が自動的に指定されますので、通常指定する必要はありません。

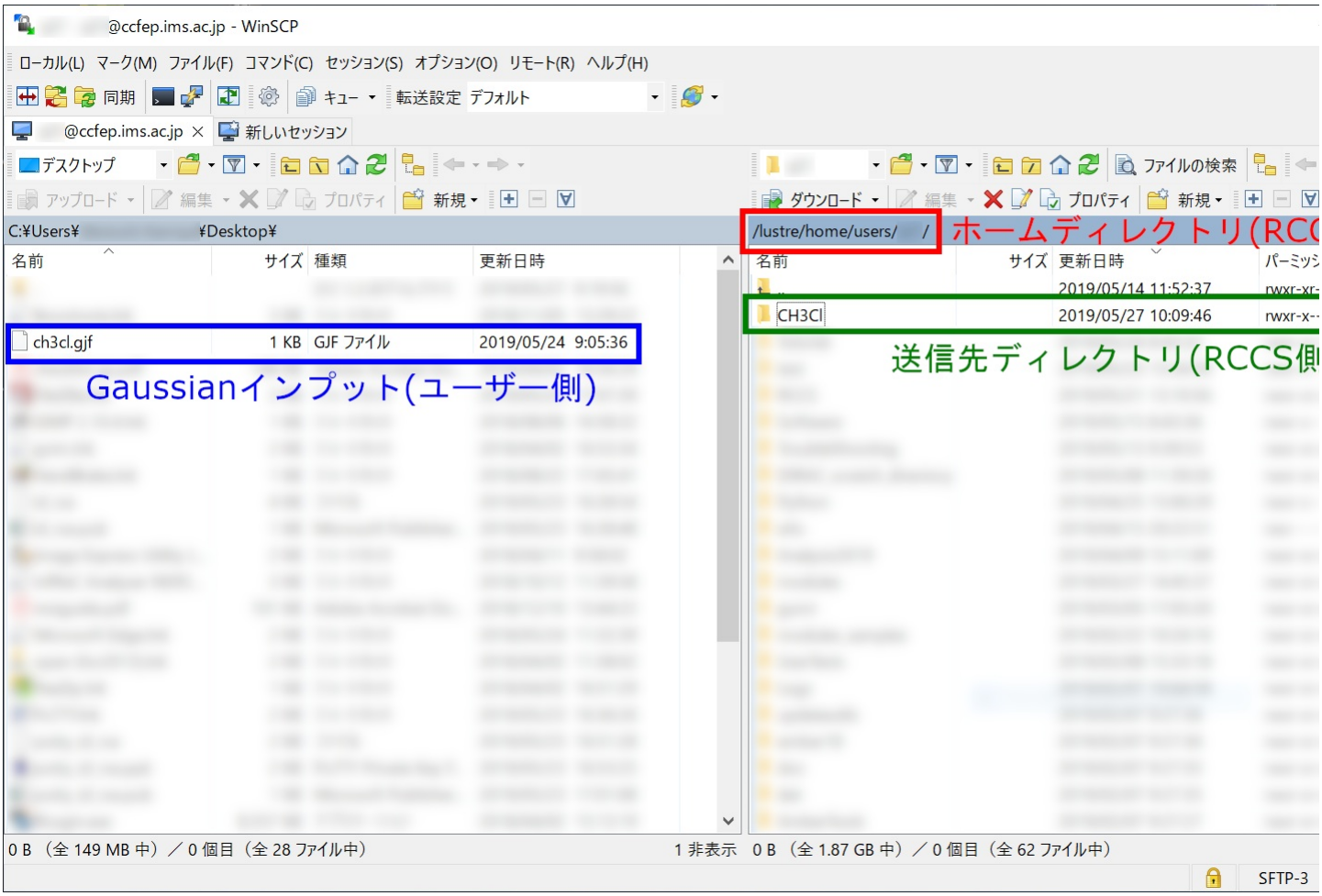
ファイルの転送(1)

WinSCP での例を示します。



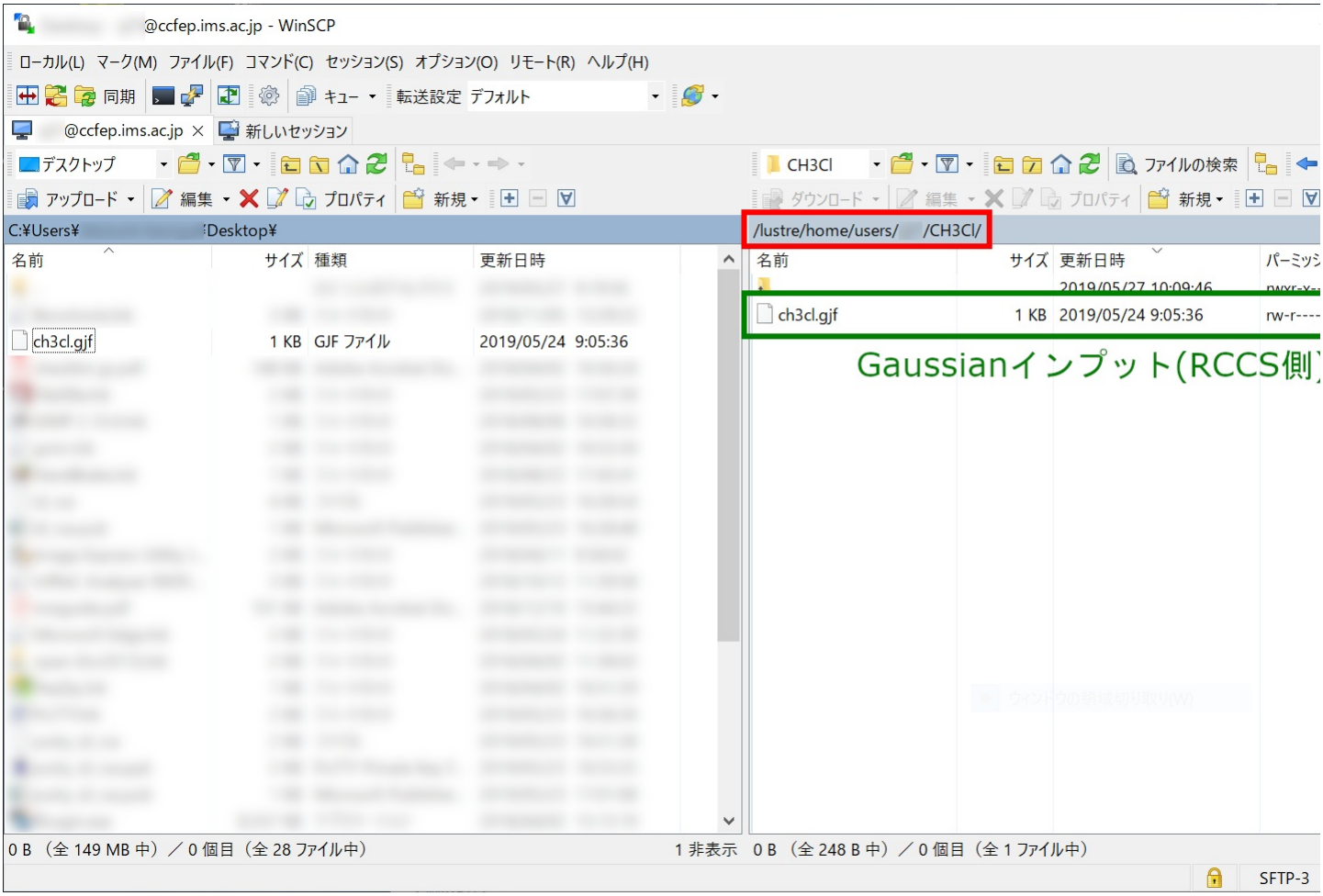
ccfep.ims.ac.jp に接続し、手元の PC 上の Gaussian のインプットを置いたフォルダに移動します。
(注意: RCCS システム上では /lustre/home/users/(ユーザー名) と /home/users/(ユーザー名) は同一の場所を指します)

ファイルの転送(2)



(必要に応じて)RCCS側にディレクトリ、ここでは RCCS 側でのホームディレクトリ直下の CH3Cl (/home/users/(ユーザー名)/CH3Cl)、を作成します。

ファイルの転送(3)



ch3cl.gjf を RCCS に転送します。以下ではこの CH3Cl ディレクトリに置いた ch3cl.gjf というインプットを実行します。
別の場所に置いた場合には以下の記述を適宜読み替えて対応ください。

g16sub でジョブを投入する

Gaussian ジョブを投入するため、ssh (PuTTY 等)で ccfep.ims.ac.jp にログインします。
ログインができたら、CH3Cl ディレクトリに cd コマンドで移動し、ls コマンドで先ほど転送したファイルがそこにあることを確認します。

```
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]$
```

そして、g16sub コマンドでジョブを投入します。

```
[user@ccfep8 CH3Cl]$ g16sub ch3cl.gjf
```

このようにオプション指定無しで g16sub コマンドを実行した場合、core の jobtype で 6 コアを利用します。
計算の制限時間(walltime)は 72 時間となっています。時間内に終わらなかった場合は強制的に終了となります。

実行すると、以下のような情報が表示されます。

- 緑の部分は投入したキューの基本情報です。コア数あたりのメモリ量等が表示されます。
- 青の部分にはジョブに関係したファイル等の名前が表示されます。
- 一番最後、赤で示した数字がジョブ ID と呼ばれるユニークな ID です。投入に失敗した場合はこの数字が表示されません。

```
[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]$
```

ジョブの状態確認

投入したジョブの状態は jobinfo コマンドで確認できます。
(投入した直後の場合は表示されないこともあります。その場合は少しお待ち下さい。)

```
[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      6  ***/---   00:00:00 cccc120
-------------------------------------------------------------------------------
[user@ccfep8 CH3Cl]$
```

- 赤文字の数字: g16sub 実行時の最後に表示されるものと同じジョブの固有 ID です
- ジョブのステータス(緑文字): Run ならば実行中で、Queue ならば他のジョブの終了を待っている状態です
- 経過時間と実行中のノード名(青文字): 時間は実行時間(Run の場合)もしくはこれまでに待った時間(Queue の場合)です。
    - Run の場合は右端のカラムに実行中のノード名が表示されます。
    - Queue の場合は実行待ちになっている理由が表示されます。
        - (cpu) 空き CPU が足りないために待っています
        - (long) 次回メンテまでにジョブが終わらないため、実行できない状態です
        - (other) 投入直後はこの表示になることがあります。

一旦ジョブが投入されてしまえば、SSH接続を切断してもジョブは残ったままになります。

ジョブが終了すると

ジョブが終了すると、実行したディレクトリには以下のようなファイルが残っているはずです。
この内、PN_ で始まるファイルと .lx で終わるファイルについては g16sub が作成したもので、正常に終了した場合には気にする必要はありません。

```
[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]$
```

出力ファイル(ch3cl.out)についてはジョブの実行中も less や tail コマンドで内容を確認できます。
今回のインプットのように %chk でチェックポイントファイルを指定していれば、それもここに作成されます。

formchk の実行

チェックポイントファイル(.chk)を formchk コマンドでテキスト形式(.fchk)に変換したい場合、まず以下のようなコマンドで設定を読み込む必要があります。

ログインシェルが /bin/bash もしくは /bin/zsh の場合:

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

ログインシェルが /bin/csh の場合:

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

上記コマンドを実行すれば、一見なにも起こりませんが内部設定が変更され、formchk コマンドが使えるようになります。

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

もし、自身のログインシェルがどちらかわからない場合は以下のように echo $SHELL コマンドを実行してください。

```
[user@ccfep8 CH3Cl]$ echo $SHELL
```

Gaussian ジョブ実行のヒント(1): g16sub のオプション

以下の二つは同一の設定になります。

```
[user@ccfep8 CH3Cl]$ g16sub ch3cl.gjf
[user@ccfep8 CH3Cl]$ g16sub -j core -rev g16c01 -np 6 -walltime 72:00:00 ch3cl.gjf
```

- -j: jobtype を指定します(デフォルト=core)。大規模なジョブを実行する場合には -j small も利用できます。
- -np: 利用するコア数を指定します(jobtype=core の場合デフォルト=6)。
- -walltime: 計算の時間制限を指定します(デフォルト 72:00:00 (72時間))。

オプションの値を変更すれば設定も変わります。例えば -np 4 とすれば 4 コアだけ利用するようになります。

Gaussian ジョブ実行のヒント(2): 別の revision の Gaussian を使う

RCCS では以下のバージョン、リビジョンの Gaussian が利用できます。

- Gaussian 16 Rev. C.01 (g16sub -rev g16c01) (g16sub デフォルト)
- Gaussian 16 Rev. B.01 (g16sub -rev g16b01)
- Gaussian 16 Rev. A.03 (g16sub -rev g16a03)

- Gaussian 09 Rev. E.01 (g09sub -rev g09e01) (g09sub デフォルト)
- Gaussian 09 Rev. D.01 (g09sub -rev g09d01)
- Gaussian 09 Rev. C.01 (g09sub -rev g09c01)
- Gaussian 09 Rev. B.01 (g09sub -rev g09b01)

Gaussian ジョブ実行のヒント(3): コア数の指定について

- (jobtype=core を想定した場合のヒントです。TCP Linda を導入していないため、ノード間並列はできません。)
- コア数を使えば使うほど速度が上がるわけではありません。多く使うことで速度が落ちる可能性もあります。
  - CPU 点数の効率的な利用、ジョブの実行されやすさの観点でもコア数は少なめの方が効率が良いです。
  - 一方で、コア数を抑えすぎると計算がなかなか終わりません。
  - 適切な数値は状況にも依存します(論文の revise でできるだけ早く結果を出せなければいけない場合と、数日待っても良い場合では最適解は違います)

Gaussian ジョブ実行のヒント(4): 作業ディレクトリについて

作業ディレクトリ(SCRATCH)は g16sub を実行した瞬間では確定していません。以下のような表記になっているはずです。(**** の部分にはあなたのユーザーIDが入ります)

```
SCRATCH dir: /work/users/****/${PBS_JOBID}
```

PBS_JOBID の部分には上で述べたジョブの固有 ID (上の例ならば 4529602.cccms1)が入ります。
作業ディレクトリの場所は出力ファイル(上の例では ch3cl.out)の冒頭でも確認できます。
なお、g16sub で投入したジョブが正常に終了した場合、この作業ディレクトリは自動的に削除されます。

このような挙動を変えたい場合は下の項目を参照ください。

Gaussian ジョブ実行のヒント(5): g09sub/g16sub を使わない方法

何らかの方法でジョブのテンプレートを作成する必要があります。ジョブスクリプトさえ作成できれば、jsub コマンドで投入するのみとなります。

- /local/apl/lx/g16c01/samples 以下(Gaussian 16 C.01 の場合)にあるサンプルをテンプレートとして作成
- g16sub を -P オプションつきで実行して生成した PN_***** ファイル(* には数字が入ります)をテンプレートとして、適宜修正

どちらの場合でも、完成したジョブスクリプトファイルを jsub コマンドで投入することになります。