Gaussian ジョブの投入方法(g16sub, g09sub)

(最終更新日: 2024/11/19)

RCCS では Gaussian ジョブ向けに g16sub, g09sub という専用コマンドを用意しています。
クイックスタートガイドにも g16sub, g09sub についての情報があります。

実行方法

Gaussian のインプットファイル(mol.gjf)を用意し、そのファイルがあるディレクトリ(somewhere)に cd コマンドで移動した上で g16sub mol.gjf と入力して実行します。メモリや利用するコア数(%mem, %nprocshared, %cpu)については g16sub が自動的に設定します。ファイル中に情報が存在していた場合は g16sub に上書きされます。それ以外の設定、例えば %chk や %oldchk などについては適宜設定してください。

```
[user@ccfep somewhere]$ g16sub mol.gjf
```

(実際に入力するのは g16sub mol.gjf の部分のみです)

オプション無しの標準設定では 8 コア、9.6 GB のメモリ(%mem)、72 時間の時間制限で実行されます。利用可能なメモリ量は CPU コア数から決定されます。メモリを多く使いたい場合は利用する CPU コア数を増やしてください。

実行例

実行すると以下のような出力が得られ、すぐに終了します(***の部分はインプットを置いた場所などに依存します)。最後に 4008669.ccpbs1 のような表示が出て、特にエラーメッセージが出力されなければ大丈夫です。

```
$ g16sub mol.gjf
QUEUE detail
--------------------------------------------------------------------------
QUEUE(MACH)  Jobtype  MaxMem     DefMem     TimLim     DefCPUs(Min-Max)
--------------------------------------------------------------------------
  H(  ap)         1.8GB      1.2GB      72:00:00   8(1-128)
--------------------------------------------------------------------------
JOB detail
==========================================================================
MOL name(s)    : mol
INP file(s)    : mol.gjf.ap
OUT file(s)    : mol.out
Current dir    : /lustre/home/users/***
SCRATCH dir    : /lwork/users/${USER}/${PBS_JOBID}/gaussian

QUEUE          : H
Memory         : 9.6GB
Time limit     : 72:00:00
Job script     : /lustre/home/users/***/H-1571896.sh
Input modified : y
==========================================================================

/usr/local/bin/jsub -q H /lustre/home/users/***/H-1571896.sh

4008669.ccpbs1
$
```

上記のようにオプション無しで実行した場合には 8 CPU コアを利用し、72 時間の制限で実行されます。上記の出力中にもそれらの情報が出力されています。

g16sub/g09sub のオプション

以下のオプションが g16sub では利用可能です。g09sub でも同様に指定できます。

--help

オプションの一覧を表示することができます。以下に説明していないオプション(基本的に使用頻度の低いものです)もいくつか存在しています。

-np (CPU コア数)

利用する CPU コア数を指定します。指定しない場合は 8 コアです。1-64 と 128 が指定できます。ここで指定された数字に合わせてインプットファイル中の %Mem に与える値が自動的に決定されます。

--walltime (時間)

計算時間を指定します。時間は (時間):(分):(秒) の書式です。
デフォルトは 72:00:00 (72 時間)です。

-j (jobtype)

ジョブタイプの指定。core, vnode, gpu, largemem から選択可能。largemem 以外の場合は自動的に判定されるため指定は不要です(コア数 1-63 ならば "core", 64, 128 ならば "vnode", GPU を 1 枚以上使うのであれば "gpu" となります)。largemem ノードを使いたい場合だけは明示的に指定する必要があります(-j largemem)。

-N

通常はスクラッチ領域として高速な計算ノードのローカルディスクの /lwork/users/${USER}/${PBS_JOBID} が使われますが、このオプションを指定すると低速ですが大容量の /gwork/users/${USER} 以下にスクラッチ領域が作られます。/lwork では容量制限で実行できない場合にご利用ください。

-M

ジョブの実行開始時、終了時にメール通知するようにします。

-rev (revision 名)

Gaussian の revision を指定します。以下の revision が利用可能です。

- g16b01 (Gaussian 16 Revision B.01)
- g16c01 (Gaussian 16 Revision C.01)
- g16c02 (Gaussian 16 Revision C.02; デフォルト)

-P

インプットとジョブスクリプトの準備までだけ行います。
ジョブは投入されません。

-ng (GPU 数)

利用する GPU 数を指定します。最大で 8 となります。(CPUコア数)/(GPU数) <= 16 でなければいけません。

--name (ジョブ名)
-C (ジョブ名)

ジョブに名前をつけます。ジョブの名前は jobinfo 実行時に表示されます。--name と -C は全く同じ機能です。使いやすい方をご利用ください。

--autoname
-X

ジョブに自動的に名前をつけます。インプットファイルからディレクトリ名と拡張子を除いたものが名前になります。--autoname と -X は全く同じ機能です。使いやすい方をご利用ください。

-O

ファイルを上書きします。

-q (キュー名)

投入するキューの指定。デフォルトで H キューが指定されているため通常は指定不要です。

一例として、以下の二つは同一の設定になります。

```
$ g16sub ch3cl.gjf
$ g16sub -q H -j core -rev g16c02 -np 8 -walltime 72:00:00 ch3cl.gjf
```

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。

```
$ g16sub -np 4 ch3cl.gjf
```

ヒント

作業ディレクトリについて

標準では、作業ディレクトリは計算サーバー上のローカルディスク上の /lwork/users/${USER}/${PBS_JOBID} 以下に作成されます。/lwork の使用可能な容量は利用するコア数に比例します(11.9 GB/コア)。容量が足りない場合は、利用コア数を増やすことで問題が解消する可能性があります。計算サーバの /lwork は高速ですが、容量は大きくないため、ジョブの種類によっては最後まで実行できないかもしれません。そのような場合は g16sub に -N オプションをつけて実行してください。低速ではありますが、容量無制限の /gwork/users/${USER}/${PBS_JOBID} を使うことができます。(ただし、ジョブの終了後にシステム側に削除される場合があります。また、定期メンテナンス時にも削除されます。)

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。

```
SCRATCH dir: /lwork/users/${USER}/${PBS_JOBID}/gaussian
```

ジョブの実行時に ${USER} はユーザー名、${PBS_JOBID} がジョブ ID (上の例ならば 4008669.ccpbs1)に置き換えられます。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、/lwork 以下はジョブの実行中のみ有効なディレクトリであるため、このスクラッチはジョブの終了と同時に削除されます。

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)
- メモリや /lwork の容量不足でエラーとなった場合は、コア数を増やすことで解決する可能性があります。
  - (エラーメッセージがわかりにくい場合もありますが、そのような場合でも一度コア数を増やして再実行しても良いかもしれません。)

g16sub/g09sub を使わない方法

Gaussian の実行の前後になにか特殊な操作を入れたい場合や、特殊な設定が必要な場合には g16sub/g09sub を使わず、jsub コマンドでジョブを投入する必要があるかもしれません。そのような場合は、以下の方法などでジョブスクリプトのテンプレートを用意することができます。

- /apl/gaussian/16c02/samples 以下(Gaussian 16 C.02 の場合)にあるサンプルをテンプレートとして利用
- g16sub を -P オプション付きで実行し、生成した H_(数字).sh ファイル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。スクラッチディレクトリ内の情報(例えば .rwf ファイル)を確保したい場合は、スクリプトファイル中で cp コマンドで確保するか、あるいは /lwork/users/$USER ではなく /gwork/users/$USER 以下を作業ディレクトリに使うなどする必要があります。/gwork 以下もいずれは自動で削除されますが、/lwork ほど速やかに消去されないため、ジョブの終了後に手動でファイルをコピーする余裕はあると思われます。(ただし、定期メンテナンスのタイミングにはご注意ください。)