付録Ｃ　　複数プログラムの実行方法と環境変数について

2007 年 4 月より SR11000/J2 の並列用ジョブキューの大部分が 1 ノード 16CPU として運用されています。現在、1 ノード 16CPU のキューと 1 ノード 8CPU のキューは下記のようになっています。

1 ノード 16 CPU のキュー　：　parallel(P001～P064), debug, Q001, Q004, Q008
1 ノード　8 CPU のキュー　：　S1, S4, H1

1 ノード 16CPU のキューでは 1 ノード 8CPU のキューに比べて 2 倍の CPU が利用できますが、16CPU が有効に利用されるような並列化が行われていなければ、CPU リソースの無駄使いとなってしまいます。このような場合、無理に 16 スレッド並列で実行させるよりも、低めのスレッド数に設定して、入力データが異なるような複数のプログラムを同時に走らせる方が効率がいい場合もあります。ここではこのような場合の使い方を紹介します。

1. 16 個の逐次プログラムを実行する場合

並列化をしない逐次プログラムの場合、1 ノード 16CPU のキューでは同時に 16 個までほとんど速度低下なく実行することができます。prog1 から prog16 までの 16 個のプログラムを走らせる場合には、次のような NQS スクリプトになります。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
./prog1 &
./prog2 &
・・・・・
./prog15 &
./prog16
wait
```

最後の wait コマンドはバックグラウンドで実行されている全てのプロセスが終了するまで待機するコマンドです。この wait が無ければ、prog1 から prog15 が実行中であっても prog16 が終了したらジョブの実行は終了してしまいます。wait の代わりに sleep コマンドを用いて一定時間待つという方法もありますが、wait コマンドを使えば、全てのプログラムが終了した時点でジョブを終了できるので無駄がありません。

なお、1 ノードの CPU の数よりも多くのプログラムを同時に実行させると、効率が悪くなります。そのため、1 ノード 8CPU のキューを利用する場合、同時に走らせるプログラムは最大 8 個にしましょう。

この方法は並列実行コマンド(prun コマンド)を利用しても実現できます。prun コマンドの定義ファイルを用いることで、さまざまな方法でプログラムを並列実行させることが可能となります。ただし、1 ノード上で複数のプログラムを実行する場合には、NQS スクリプト中に「#@$-J T16」あるいは「#@$-J SS」の指定が必要となりますので注意してください。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
#@$-J T16
prun -f sample.def
```

prun 用定義ファイル(sample.def)

```
*16 ./prog%n
```

prun コマンドは本センターSR11000/J2 上において独自に提供されているコマンドです。利用方法の詳細については、オンラインマニュアル(「% man prun」を実行)、あるいは、「11.1 並列実行(prun コマンド)」をご覧ください。

2. 4 個の要素並列化プログラムを実行する場合

要素並列化プログラムの並列実行スレッド数は、特に指定しなければ実行するキューにおける 1 ノードの CPU の数と同じになります。しかし、高い並列性がないプログラムでは、並列実行スレッド数を上げてもそれに比例した速度向上は期待できません。この場合、並列スレッド数を低く設定して、複数のプログラムを同時に実行するといった方法も考えられます。4 個の要素並列化プログラムをそれぞれ並列実行スレッド数 4 で実行する場合の NQS スクリプトは下記のようになります。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
setenv HF_PRUNST_THREADNUM 4
./prog1 &
./prog2 &
./prog3 &
./prog4
wait
```

環境変数 HF_PRUNST_THREADNUM に各プログラムで利用する並列実行スレッド数(1～16)を指定します。実行するプログラム毎にこの環境変数の値を変更することもできます。また、この環境変数は最適化 FORTRAN77/90、最適化 C、最適化標準 C++で共通です。

なお、並列実行コマンド(prun コマンド)を利用しても実現できます。この場合、NQS スクリプトと prun 用定義ファイルは下記のようになります。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
#@$-J T4
setenv HF_PRUNST_THREADNUM 4
prun -f sample.def
```

prun 用定義ファイル(sample.def)

```
*4 ./prog%n
```

3. 4 個の OpenMP 並列プログラムを実行する場合

4個のOpenMP並列プログラムをそれぞれ並列実行スレッド数4で実行する場合のNQSスクリプトは下記のようになります。

NQSスクリプト

```
#@$-q parallel
#@$-N 1
setenv OMP_NUM_THREADS 4
./prog1 &
./prog2 &
./prog3 &
./prog4
wait
```

環境変数OMP_NUM_THREADSにはOpenMPで利用する並列実行スレッド数を設定します。もし、環境変数HF_PRUNST_THREADNUMが設定されている場合、環境変数OMP_NUM_THREADSの値は、環境変数HF_PRUNST_THREADNUMの値以下でなくてはいけません。環境変数OMP_NUM_THREADSが設定されていない場合には、環境変数HF_PRUNST_THREADNUMの値とみなされます。

なお、並列実行コマンド(prunコマンド)を利用しても実現できます。この場合、NQSスクリプトとprun用定義ファイルは下記のようになります。

NQSスクリプト

```
#@$-q parallel
#@$-N 1
#@$-J T4
setenv OMP_NUM_THREADNUM 4
prun -f sample.def
```

prun用定義ファイル(sample.def)

```
*4 ./prog%n
```

4. 複数ノードで複数の並列化されたプログラムを実行する場合

例えば、4ノードを用いて8個の並列化されたプログラムを実行する場合、1ノード16CPUのキューであれば1つの並列プログラムに8CPUを割り当てることができます。このような場合には、並列実行コマンド(prunコマンド)を利用します。4ノードで8つのプログラムprog1、prog2、・・・、prog8を実行する場合のNQSスクリプトとprun用定義ファイルは下記のようになります。

NQSスクリプト

```
#@$-q parallel
#@$-N 4
#@$-J T2
setenv HF_PRUNST_THREADNUM 8
prun -f sample.def
```

prun用定義ファイル(sample.def)

```
*4 ./prog%n
```

NQSスクリプト中「#@$-J T2」の部分は、1ノードあたり2つのプログラムを実行することを指定しています。「#@$-J T4」のように間違って設定してしまうと、1ノードに4つのプログラムが実行され、2ノードだけで8つのプログラムが実行されてしまうことになるので、注意が必要です。

5. MPIプログラムで異なる実行モジュールを起動する方法

MPI プログラムでは、各プロセスのランク番号毎に異なる実行モジュールを起動させることもできます。各システム固有の方法がありますが、ここでは少し汎用的な方法を紹介します。

1 ノード 4MPI プロセスを利用する場合には、下記のような NQS スクリプトを記述します。「#@$-J T4」のところで mpirun コマンドで起動するプロセス数を指定します。1 ノード 16CPU のキューで 4MPI プロセスを実行する場合、各プロセスは 4CPU まで占有できます。そこで、環境変数 HF_PRUNST_THREADNUM には 4 を設定します。この環境変数の設定を行わないと 16 が仮定されますので、注意が必要です。また、OpenMP 並列プログラムでは、環境変数 OMP_NUM_THREADS を 4 に設定しても構いません。ここでは、mpirun の引数に./exec.sh を与えて、exec.sh を 4 つ実行するようにします。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
#@$-J T4
setenv HF_PRUNST_THREADNUM 4
mpirun ./exec.sh
```

exec.sh の中身

```
#!/bin/sh
if [ "$MP_CHILD" == "0" ]; then
  exec ./mpi_prog1
fi
if [ "$MP_CHILD" == "1" ]; then
  exec ./mpi_prog2
fi
if [ "$MP_CHILD" == "2" ]; then
  exec ./mpi_prog3
fi
if [ "$MP_CHILD" == "3" ]; then
  exec ./mpi_prog4
fi
```

SR11000/J2 では MPI プロセスを起動する際、環境変数 MP_CHILD には各プロセスのランク番号が設定されます。そこで、環境変数 MP_CHILD の値を見て、どのプログラムを実行するかを決定しています。この例では、ランク番号の小さい順に、mpi_prog1、mpi_prog2、mpi_prog3、mpi_prog4 を実行しています。この環境変数 MP_CHILD の名前は MPI の実行環境によって異なりますので、他のシステムで利用する際には変更が必要になります。

なお、並列実行コマンド(prun コマンド)を利用すると、もっと簡単になります。上記と同じように 1 ノード 4MPI プロセスで、ランク番号毎に異なる実行モジュールを起動させたい場合には、NQS スクリプトと prun 用定義ファイルは下記のようになります。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
#@$-J T4
setenv HF_PRUNST_THREADNUM 4
prun -f sample.def
```

prun 用定義ファイル(sample.def)

```
*4 ./mpi_prog%n
```

4 ノードで 4MPI プロセスを実行したい場合には、NQS スクリプト中の「#@$-N 1」を「#@$-N 4」に変更し、「#@$-J T4」と「setenv HF_PRUNST_THREADNUM 4」は削除してください。

6. スレッドを CPU に固定的に割り付ける方法

環境変数 HF_PRUNST_BIND の設定で、スレッドを CPU に固定的に割り付けるかどうかを指定します。固定的に割り付けた場合、各スレッドは常に割り付けられた番号の CPU 上で実行されます。

setenv HF_PRUNST_BIND 0　・・・　スレッドを CPU に固定的に割り付けない
setenv HF_PRUNST_BIND 1　・・・　スレッドを CPU に固定的に割り付ける

この環境変数を設定しない場合には、0 を設定したとみなされます。この環境変数が有効となるのは、コンパイル時に -parallel オプションで要素並列化されたプログラム、および、-parallel -omp オプションで OpenMP 並列化が行われたものに限ります。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
setenv HF_PRUNST_BIND 1
./prog1
```

この例では、要素並列化あるいは OpenMP 並列プログラムである prog1 の各並列実行スレッドをそれぞれ特定の CPU で実行されるよう固定的に割り付けます。

ただし、この環境変数の設定による速度向上は実行するプログラムによって大きく異なり、速度向上がほとんどない場合もあります。また、独自にマルチスレッドプログラミングを行ったプログラムの場合は、環境変数 HF_PRUNST_BIND の設定では固定的に割り付けることはできません。プログラム中で bindprocessor システムコールを用いてください。また、一般のプロセスを固定的に割り付ける場合には bindprocessor コマンドを用いてください。これらの詳細はオンラインマニュアル(「% man bindprocessor」や「% man 2 bindprocessor」を実行)などでご確認ください。

7. スレッドの待機方法の指定

環境変数 HF_PRUNST_WAIT の設定で、スレッドの待機方法を指定することができます。

setenv HF_PRUNST_WAIT BUSY　・・・　待機スレッドはスピンループを実行して待機する
setenv HF_PRUNST_WAIT SLEEP　・・・　待機スレッドはスリープして待機する

この環境変数を指定しない場合には、BUSY を設定したとみなされます。スピンループで待機している場合、次の仕事が来た場合に即座に開始できます。しかし、スピンループをしている間は余計な CPU 時間を使っていることになります。1 ノード 16CPU のキューで、各プロセスの並列実行スレッドの合計数が 16 以下であれば問題ありませんが、16 を超える可能性がある場合には、必ず「setenv HF_PRUNST_WAIT SLEEP」の設定をしておきましょう。

NQS スクリプト

```
#@$-q parallel
#@$-N 1
setenv HF_PRUNST_WAIT SLEEP
setenv OMP_NUM_THREADS 16
./prog1 &
./prog2
wait
```

OpenMP 並列プログラム prog1 と prog2 を同時に実行する場合の例です。要素並列化プログラムの場合には、「setenv OMP_NUM_THREADS 16」の代わりに「setenv HF_PRUNST_THREADNUM 16」とします。