新スーパーコンピュータ（HPC2500）利用のしおり

津　田　知　子

Ⅰ．はじめに

前号の本センターニュースに掲載された「新スーパーコンピュータシステムの概要と利用方法について」や「ベクトル（VPP5000）からスカラSMP（PRIMEPOWER HPC2500）へ」の紹介記事で，新スーパーコンピュータの大体の概略はつかんでいただけたと思います。ところが，実際に自分のプログラムをHPC2500で流してみようとすると，スレッド並列のやり方，プロセス並列でのメモリ量やCPU時間の指定方法，プログラムのチューニング方法等など詳細な点でいろいろわかりにくい部分があるようで，たくさんの質問が相談窓口に寄せられました。今回は，そんな寄せられた質問を基に，新スーパーコンピュータでプログラムを作成し，計算したい利用者のための利用方法を紹介することにします。

Ⅱ．新スーパーコンピュータHPC2500の特徴とその利用方法

新システムでは，スーパーコンピュータとアプリケーションサーバを一体的に運用し，hpcシステムと呼んでいます。以下にhpcシステムの特徴を列挙します。

・ノード数：24
・CPU台数：1664
・１CPUの理論最大性能：８Gflops
・総合理論性能：13Tflops
・１ノードのメモリ容量：512GB
・総メモリ容量　12TB
・クロスバネットワークのデータ転送速度：最大４GB／秒
・ディスク容量　100TB
・分散並列スカラ計算機

これらのことから，

➢ CPUがたくさんある。
➢ メモリ量が潤沢である。
➢ CPUのマシンクロックが速い。
➢ クロスバネットワークのデータ転送速度が速い。

といった特徴を大いに活かしてhpcシステムの利用を考えてみてください。

プログラムから見たhpcシステムでの処理形態は，図１に示すように分類することができます。

多くのCPUを活かすという点では，並列処理がお勧めです。この並列処理の効果は，プログラムの処理経過時間が短縮できることです。CPU時間は，プログラムを逐次実行（１CPUで実行）した場合のCPU時間と同じか，または，並列処理のオーバヘッド分だけ増加したものになります。他のジョブとのCPUの競合が起こった場合には，並列処理の効果を得ることができないどころか，CPU時間も増大していきます。これは，他のジョブでの実行が終了するまで，CPU資源が確保されそれがCPU時間として計上されるためです。つまり，並列処理で効果を得るためには，CPUの競合がおきない環境で処理することが前提となります。並列ジョブは，TSSではなく，NQSによるバッチ処理で実行してください。利用負担金の面からもバッチ処理が，お勧めです。

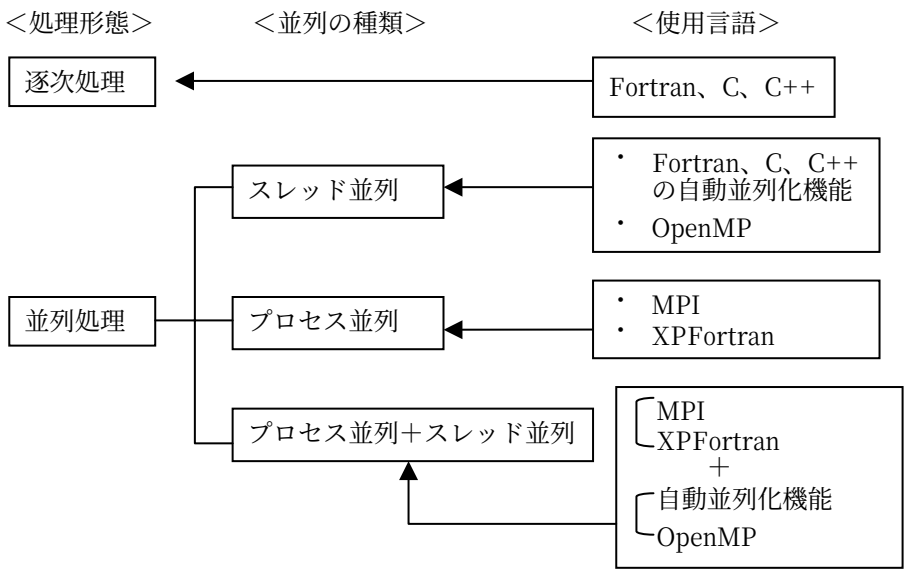

図１　hpcシステムでの処理形態

Ⅲ．処理形態と使用例

１．逐次処理

Fortran（frtコマンド），C（fccコマンド），C++（FCCコマンド）の言語により翻訳処理して，実行します。実行時間が長い場合には，NQSによるバッチ処理がお勧めです。使用できるメモリ容量は，１プロセスあたり最大400GBです。

＜メモリ２GBを超える１CPU用のモジュールをNQSで実行＞

（１）翻訳処理

```
hpc% frt  -o  test_s5g  test_s5g.f
```

（２）NQSで実行するためのスクリプトファイルの作成

実行キューとしてp8を指定し，qsubの-lMオプションで必要メモリ量を指定する。

スクリプトファイル　nqs/exec_s5g.sh

実行可能ファイル名：pg/test_s5g

```
#  @$-q p8  -eo  -o test_s.out  -lM 5gb
cd  pg
./test_s5g
```

（3）1CPU用モジュールの実行依頼

スクリプトファイル：exec_s5g.sh

qsubコマンドで依頼する。

```
hpc% qsub exec_s5g.sh
```

2．並列処理（スレッド並列）

hpcシステムのFortran，C，C++のコンパイラには，自動並列化の機能が備わっています。コンパイルオプションで，-Kparallelと指定するだけでプログラムは，簡単に自動並列化されます。例えば，Fortranのプログラムでは，多くの場合DOループの外側が，スレッド並列という形で並列化されます。各スレッドに別々のCPUを割り当てて実行することにより，プログラムの実効性能をあげることが期待できます。図2にFortranでのスレッド並列処理の例（2CPUで実行）を示します。

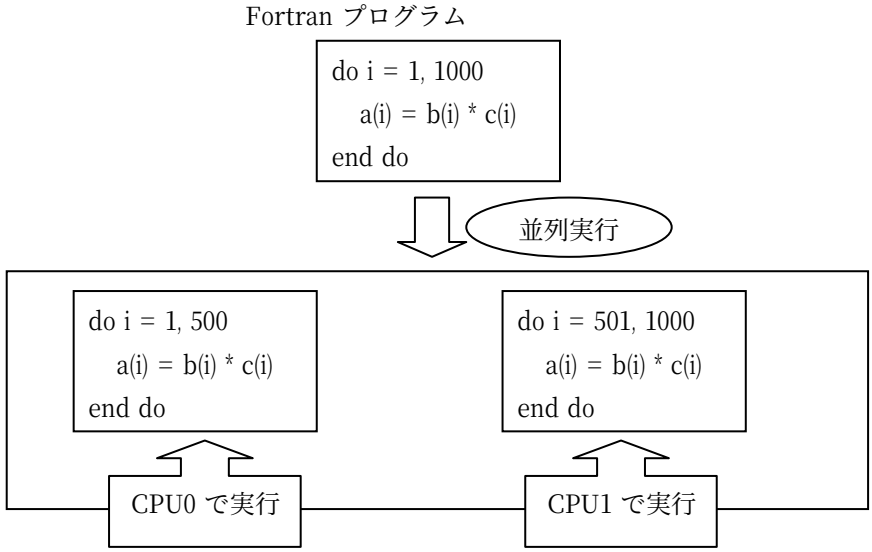

図2　Fortranプログラムでのスレッド並列処理の例

しかし，その並列化による効果は，プログラムに大きく依存しますので，現在お持ちのプログラムを一度自動並列化して実行してみることをお薦めします。なお，自動並列化だけでは，性能を向上させることができない場合には，利用者自身で，OpenMPディレクティブを挿入することにより更なる並列化を行うことも可能です。

＜スレッド並列モジュールの実行例＞

frtコマンドの-Kparallelオプションを指定して自動並列化を行ったり，OpenMPを用いて並列化した場合には，スレッド並列モジュールとなります。スレッド並列モジュールの作成は，TSSで行い，実行は，NQSによるバッチ処理をお勧めします。

実行に際しては，スレッド並列モジュールをいくつのCPUで実行するかが，大きな問題です。残念ながら推奨のCPU数があるわけではありません。各自のプログラムに依存しますので，４CPUまたは８CPUでプログラムを実行し，その実効性能を見ながら，CPU数を増減させる必要があります。各自のプログラムにとってもっとも実効性能の良い最適なCPU数を見つけてください。

（１）翻訳処理

```
hpc% frt  -Kparallel  -o tpara  tpara.f
```

（２）実行のためのスクリプトファイル作成

ここでは，スレッド並列モジュールを８CPUで実行する場合の例を示します。実行キューは，p8（8CPUまで利用可能）を指定します。スレッド数の指定は，-lpオプションで行います（ここで，pは小文字)。使用するメモリが2GBを超える場合には，-lMオプションで必要メモリ量を指定します。また，-lTオプションでこのプログラムで使用するCPU時間（この例では12時間30分）を指定しています。指定しない場合には，10時間で実行が打ち切られます。なお，-lTオプションで指定するCPU時間は，各CPUの総合計を指定します。

スクリプトファイル　nqs/exec_tp.shの内容

実行可能ファイル名：pg/tpara

```
#  @$-q p8  -lp 8  -eo  -o tpara.out
#  @$-lT 12:30:00   -lM 5gb
cd  pg
./tpara
```

（３）スレッド並列モジュールの実行依頼

スクリプトファイル：exec_tp.sh

qsubコマンドで依頼する。

```
hpc% qsub exec_tp.sh
```

３．並列処理（プロセス並列　－MPI－）

（１）翻訳処理

Fortran　mpifrtコマンド

C　　mpiccコマンド

```
hpf%  mpifrt -o mpi_prog  mpi_prog.f
```

（2）実行

hpcシステムでのMPIの実行は，mpiexecコマンドを用います。実行は，CPU競合をさけることができるバッチジョブとして，処理することをお勧めします。mpiexecコマンドのオプションでは，プロセス数，実行モード，実行可能プログラムのパス名等を指定します。mpiexecコマンドの形式をつぎに示します。

```
mpiexec  -n NP -mode MODE prog
```

ここで，

NP ：最初に静的に生成するユーザの実行可能プログラムのプロセス数。

MODE：MPIプログラムの実行モードを指定。実行モードには，limitedとfullがある。fullモードでは，MPI2規格のすべての言語仕様のプログラムを実行することが可能である。limitedモードでは，MPI2規格の言語仕様のうち，プロセス管理及びMPI I/O機能が実行できない。このオプションの省略値は，fullである。qsubで実行する場合には，-lPに指定する値は，*NP*+1を指定すること。

prog ：最初に静的に生成される実行可能プログラムのパス名。

fullモードでの実行

バッチジョブとして動作させるには，qsubコマンドによりジョブをNQSシステムに投入します。つぎに示すようなスクリプトファイルを作成し，qsubコマンドで実行します。

スクリプトファイル名：mpi_full.sh

```
#  @$-q p8  -eo -o mpi_full.out
#  @$-lP 5
mpiexec -n 4 ./mpi_prog
```

キューの指定

CPU数の指定

プロセッサ数の指定

```
hpc% qsub  mpi_full.sh
```

limitedモードでの実行

バッチジョブとして動作させるには，qsubコマンドによりジョブをNQSシステムに投入します。つぎに示すようなスクリプトファイルを作成し，qsubコマンドで実行します。

スクリプトファイル名：mpi_limited.sh

```
#  @$-q p8  -eo -o mpi_lim.out
#  @$-lP 4
mpiexec  -n 4 -mode limited ./mpi_prog
```

キューの指定

CPU数の指定

プロセッサ数の指定

```
hpf% qsub  mpi_limited.sh
```

４．並列処理（プロセス並列　－XPFortran－）

（１）翻訳

XPFortranの翻訳は，xpfrtコマンドを用います。

```
hpf%  xpfrt -o xprog  xprog.f
```

xpfrtコマンドのオプションの詳細は，manコマンドで確認してください。

（２）実行

XPFortranの実行には，xpfexecコマンドを用います。実行は，CPU競合をさけることができるバッチジョブとして，処理することをお勧めします。xpfexecコマンドのオプションでは，プロセス数，実行モード，実行可能プログラムのパス名を指定します。xpfexecコマンドの形式をつぎに示します。

xpfexec -vp *NP* -m *MODE prog*

ここで，

NP　：プロセッサ数を指定します。

MODE：fullまたはimitedを指定します。fullの場合には，I/Oサーバプロセスを起動し，共有装置入出力文は，I/Oサーバプロセスで実行されます。limitedでは，共有装置入出力文の実行のためのI/Oサーバプロセスは生成されず，共有装置入出力文は，入出力文を実行したリージョンのいずれかのプロセスで実行されます。このオプションの省略値は，fullです。fullモード指定で実行する場合には，qsubコマンドの-lPに指定する値は，*NP*+1を指定すること。

prog：最初に静的に生成される実行可能プログラムのパス名。

fullモードでの実行

バッチジョブとして動作させるには，qsubコマンドによりジョブをNQSシステムに投入します。つぎに示すようなスクリプトファイルを作成し，qsubコマンドで実行します。

スクリプトファイル名：xpf_full.sh

```
#  @$-q p8  -eo -o xpf_full.out
#  @$-lP 5
xpfexec -vp 4 ./xp_prog
```

→ キューの指定
→ CPU数の指定
→ プロセッサ数の指定

```
hpc% qsub  xpf_full.sh
```

limitedモードでの実行

バッチジョブとして動作させるには，qsubコマンドによりジョブをNQSシステムに投入します。つぎに示すようなスクリプトファイルを作成し，qsubコマンドで実行します。

スクリプトファイル名：xpf_limited.sh

```
#  @$-q p8  -eo -o xpf_lim.out
#  @$-lP 4
xpfexec  -vp 4 -mode limited ./xp_prog
```

キューの指定
CPU数の指定
プロセッサ数の指定

```
hpf% qsub  xpf_limited.sh
```

５．並列処理（プロセス並列＋スレッド並列）

MPIやXPFortranでのプロセス並列と自動並列または，OpenMPによるスレッド並列を組み合わせて利用する形態です。組み合わせとしては，以下に示すものがあります。

MPI ＋ { 自動並列化機能 / OpenMP / 自動並列化機能＋OpenMP }

XPFortran ＋ { 自動並列化機能 / OpenMP / 自動並列化機能＋OpenMP }

スレッドのCPU数は，並列の効果が得られないプログラムでは，大きな値を指定しても，CPU時間が増加するだけです。スレッド並列の効果を確認しながらCPU数を調整してください。

スクリプトファイル名：mpi_h.sh

```
#  @$-q p16  -eo -o mpi_h.out
#  @$-lP 4  -lp 4
mpiexec  -n 4 -mode limited ./mpi_progh
```

キューの指定
CPU数の指定
プロセッサ数の指定

```
hpf% qsub  mpi_h.sh
```

Ⅳ．hpcシステムに特有なこと

１．メモリに関すること

qsubコマンドの-lMオプションで指定する値は，以下を参考に指定してください。

（１）sizeコマンドからメモリ使用量を算出

XPFortranのラージページのメモリ使用量は，つぎのようにして算出します。

ラージメモリ使用量　＝　静的領域　＋　動的領域

静的領域　＝　sizeコマンドのbssとdataの合計

動的領域　＝　スタック域　＋　ヒープ域　＋　ライブラリ作業域

＜sizeコマンドの出力例＞

```
hpc% size  a.out
```

```
176748 + 876 + 1792600 = 1970224
  ↓       ↓        ↓
 text    data     bss
```

（2）qsubの-oiオプションで得られる情報から知る

qsubコマンドで-oiオプションを指定すると，標準出力に下記のようなバッチリクエストの統計情報が出力されます。この統計情報の“Used Resource”欄の“Total Max Large Page”に示されている値をつぎに実行するときのqsubコマンドの-lMのオプションで指定してください。

```
Allocated Resource
   Process Resource Set   :     1 Set
   Total Large Page       :  4098 Mbyte
   CPUs                   :     4 CPU
            :
Used Resource
   Total System CPU Time  :  1290 msec
   Total User CPU Time    :  32116540 msec
   Total Max Large Page   :  1024 Mbyte
            :
```

２．ノード間バリアとDTUについて

hpcシステムで，非常に大きなプロセス並列を行う場合，ノードにまたがるプロセス間で同期をとるためのノード間バリア資源とプロセス間の高速なデータ転送を実現するユーザDTU（Data Transfer Unit）資源が不足する可能性があります。ノード間バリア資源は，システム全体で255，ユーザDTU（UDTU）は，ノードあたり31です。他のジョブとの競合状態にもよりますが，これらの資源が不足すると予想される場合には，ノード間バリアについては，-lB0を，ユーザDTUについては，-ldt 0をqsubコマンドのオペランドで指定してください。これらの指定により，ノード間バリアについてはソフトバリアを，DTUについては，Shared-UDTU機能を使ってジョブの実行が行われます。Shared-UDTU機能については，データ転送サイズが64KB程度と小さい場合には，性能が劣化しますが，1MB以上の場合には，ユーザDTUとほぼ同じ転送性能が得られます。

Ⅴ．hpcシステムでのチューニング

hpcシステムでのチューニングは，プロファイラ情報を基に行います。

１．翻訳

-Ktl_trtオプションを指定して，翻訳します。

２．実行

以下の環境変数を設定して，実行します。環境変数の値の詳細は，マニュアル『プログラミング支援ツール使用手引書』を参照してください。

```
setenv  TRT_ENV "PMP=on"   :  プロファイラ情報収集の指定
setenv  PROF_STATS 7       :  ハードウェアモニタ情報，サンプリング情報の測定
                              （ここでは，以下の情報を指定）
                              1 Communication statistics
                              2 Performance statistics
                              4 Executing profile
setenv  PROF_PA  sta,cac   :  ハードウェアモニタ情報の測定項目を指定
                              sta  実行の統合情報
                              cac  キャッシュ利用状況
```

＜自動並列のプログラムのプロファイラ情報の取得例＞

```
#  @$-q p8 -lp 5  -eo -o proflist
cd chk
frt  -Ktl_trt  -Kparallel  -o chkprof  chkprof.f
setenv  TRT_ENV "PMP=on"
setenv  PROF_STATS 7
setenv  PROF_PA  sta,cac
setenv  PARALLEL  4
./chkprof
```

スレッド並列のプログラムのプロファイラ情報を取得する場合には，並列数を環境変数PARALLELで指定し，qsubコマンドの-lpでは，スレッド並列数＋１の値を指定すること。（この例では，PARALLELを4に，-lp 5としている）

３．プロファイラ情報の出力

２．の実行により，カレントディレクトリに以下に示すファイルが作成されます。

GProf_xxxx.prof.pri

DProf_xxxxx.000.prof.pri　～　DProf_xxxxx.nnn.prof.pri

これらのファイルをmpprofコマンドにより解析することにより，プロファイラ情報が取得できます。

-samp　：　サンプリング情報

-pa　　：　ハードウェアモニタ情報

```
hpc% mpprof  -samp  -pa  GProf_XXXXXX.prof.pri  > proflist
```

４．プロファイラ情報とチューニング

３．で得られたプロファイラ情報は，“ ヘッダ ”, “ サンプリング情報（手続きコスト，ループコスト，行コスト）”, “ ハードウェアモニタ情報 ” などから構成され，その出力は結構な量になります。この出力リストでチューニングの情報としては，まず，サンプリング情報の “ *** Global Profile ” から，各ルーチンのコスト（Sampling Cost）と負荷バランス（Parallel Balance）を見ます。プログラムの性能改善のためには，一番コストの掛かっているルーチンから手を付けていきます。また，並列処理では，なるべく各プロセスまたはスレッド間で偏りがないようにします。

```
Execution Profile
*** Global Profile

              Sampling Cost
  ----------------------------------------
  Samples       Run(%)   Barrier    Start  End
  ----------------------------------------
  1.924000e+03  68.84  0.000000e+00  59     60     diffus._PRL_2_
  5.320000e+02  19.03  5.270000e+02  41     73     diffus_
  2.020000e+02   7.23  0.000000e+00  68     69     diffus._PRL_1_
                                :
```

図３　プロファイラ（サンプリング情報）の出力例

つぎに，ハードウェアモニタ情報の “ Hardware Monitor ” で示される “ Statistics Performance ” の項目を見ていきます。ここでは，“ MFLOPS（浮動小数点演算命令実行速度）”, “ L2-miss(%)（２次キャッシュのミスヒット率）”, “ mTLB-op(%)（データメインTLBのミスヒット率）” の値に注目してください。これらの値は，MFLOPSは600以上，L2-missは，1.0以下，mTLB-opは，0.01以下が目安とのことです。これらの値から大きくかけ離れている場合には，チューニングが必要になります。

Ⅵ．おわりに

新スーパーコンピュータの利用をプログラムの面から眺めてきました。チューニング情報の取得についても簡単に触れておきましたが，これらの情報から実際のプログラムの性能を改善することは，結構大変な作業です。VPP5000に比べて性能が出ないなど性能に問題がある場合は，遠慮なく，センターまでお申し出ください。この稿が，利用の一助になれば幸いです。ユーザにとってもセンターにとっても新スーパーコンピュータが使い易い良いシステムになり，大いに利用されることを願っています。

（つだ　ともこ：名古屋大学情報連携基盤センター学術情報開発研究部門）