スーパーコンピュータ SX-7, SX-7C の利用法

情報部情報基盤課 システム管理係 システム運用係
情報シナジーセンター スーパーコンピューティング研究部

はじめに

スーパーコンピュータはSX-7Cと既存のSX-7から構成されます。SX-7Cは複数ノードによる高速な演算が可能になります。また、バッチ処理におけるジョブ管理システムはNQSからNQSⅡに変わります。この資料では、これらSX-7,SX-7Cの利用法について説明します。

システム構成

システムは、スーパーコンピュータSX-7, SX-7Cと並列コンピュータTX7/i9610から構成されます（図1）。

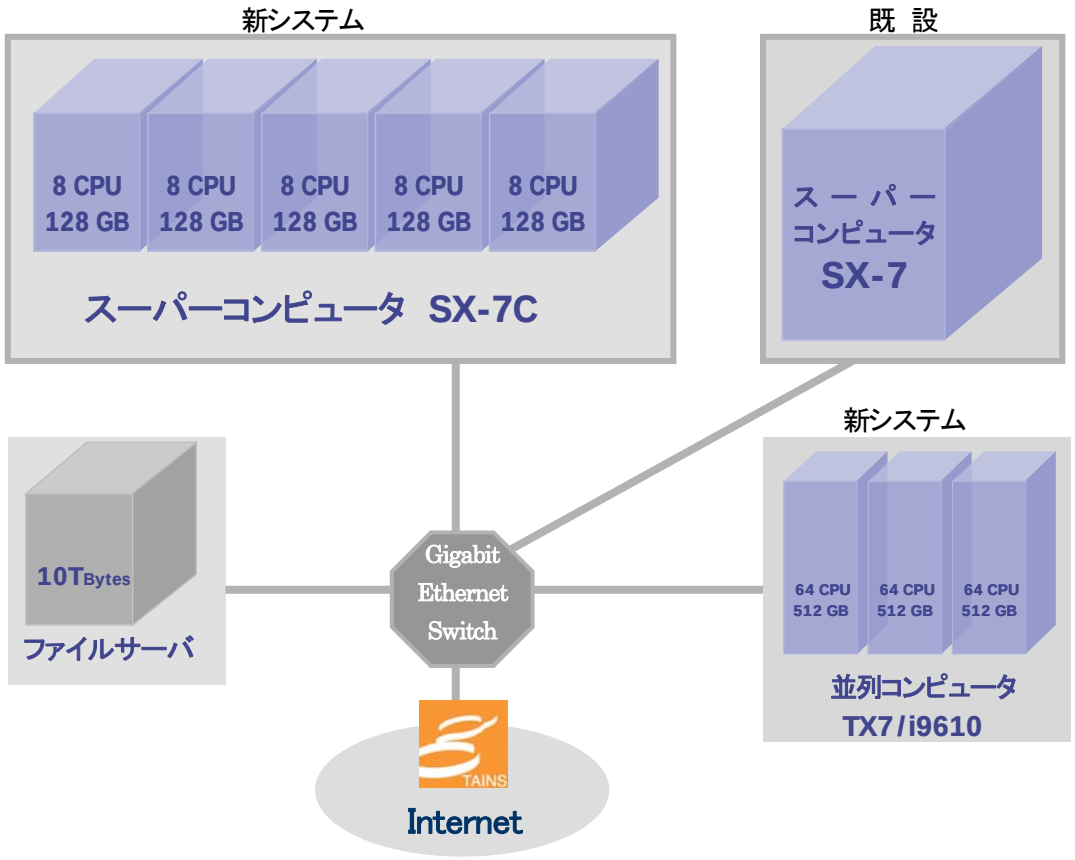

図1 システム構成

スーパーコンピュータ SX-7C

スーパーコンピュータ SX-7C は5つのノードで構成され、それぞれのノードは 16GFLOPS のベクトルプロセッサ 8 台と 128GB の主記憶装置を搭載しています。5 つのノードで最大640GFLOPS、メモリ容量640GB の大規模かつ高性能な処理が可能になります。並列処理には MPI を使った並列プログラミングが必要になります。

スーパーコンピュータ SX-7（既設）

スーパーコンピュータ SX-7 は、8.83GFLOPS のベクトルプロセッサ 32 台と 256GB の主記憶装置を搭載したシステムで1つのノードを構成します。並列処理はノード内32並列まで可能で、自動並列化、OpenMP、MPI を使った並列処理が実行できます。

並列コンピュータ TX7/i9610

並列コンピュータ TX7/i9610 の1つのノードは、Intel 最新の 64 ビットプロセッサ Itanium2™(IA-64)を 64 台と512GB の主記憶装置を搭載しています。1CPU あたり 6.4GFLOPS の計算能力を有しますので、ノードあたりの最大演算性能は 409.6GFLOPS となります。ベクトル演算に不向きなプログラムの高速な実行が可能です。また、スーパーコンピュータ SX-7 および SX-7C のフロントエンドサーバとしての役割も担っております。

ログイン

スーパーコンピュータ SX-7,SX-7C を利用するには、フロントエンドサーバの並列コンピュータ TX7/i9610 にログインします。ログインは、SSH(Secure SHell)を利用します（リスト1）。はじめて接続する場合、コマンド入力後

```
"Are you sure you want to continue connecting (yes/no)?"
```

と問い合わせがありますので yes を入力してからパスワードを入力します。

リスト1　並列コンピュータへのログイン

```
yourhost$ ssh  gen.isc.tohoku.ac.jp -l 利用者番号
Password: パスワード

[利用者番号@gen ~]$
```

Windowsからログインする場合は、Putty等SSH対応のリモート接続ソフトをご利用ください。[1]

センターに利用登録すると、図1すべてのサーバが利用可能になります。

[1] http://www.cc.tohoku.ac.jp/service/USE/index.html「システム利用の手引き」

ログイン名とパスワードは各サーバで共通です。ログイン名は利用者番号を用い、パスワードは初期パスワードが設定されていますので yppasswd コマンドで速やかに変更してください(リスト2)。また、パスワードはセキュリティ保護のためときどき変更することをお奨めします。利用登録時のログインシェルは csh を設定しています。tcsh 等に変更したい場合は ypchsh コマンドをご利用後、再ログインしてください(リスト3)。

ホームディレクトリはファイルサーバの「/uhome/利用者番号」となります。NFS(Network File System)によるファイル管理を行っていますので、スーパーコンピュータ、並列コンピュータから共通に利用できます。

リスト2 パスワードの変更

```
gen$ yppasswd
yppasswd is deprecated, use /usr/bin/passwd instead

Changing password for x2xxx9.
Old Password: 現在のパスワード
New password: 新しいパスワード
Re-enter new password: 新しいパスワード（再度）
Changing NIS password for x2xxx9 on xxx.isc.tohoku.ac.jp.
Password changed
```

リスト3 ログインシェルの変更(tcsh への変更例)

```
gen$ ypchsh
ypchsh is deprecated, use /usr/bin/chsh instead

Changing login shell for x2xxx9.
Password: パスワード
Enter the new value, or press return for the default.
       Login Shell [/bin/csh]: /bin/tcsh
Shell changed.
```

プログラミング言語、ライブラリ

プログラミング言語および科学技術計算用ライブラリとして表1に示すものが利用できます。なお、ASL/SX はコンパイル時のライブラリ指定(-l オプション)は不要です。

表1 プログラミング言語およびライブラリ

Fortran90/SX	ISO/IEC 1539-1:1997,自動並列化, OpenMP
C++/SX	ISO/IEC 14882:1998,自動並列化, OpenMP
MPI/SX	並列処理ライブラリ
ASL/SX	Fortran90/SX 用科学技術計算ライブラリ

ファイルエディット

ソースファイルは、並列コンピュータの emacs エディタまたは vi エディタで作成します。
研究室等のパソコンにあるソースファイルを利用するには、gen.isc.tohoku.ac.jpにファイル転送してください。送り元のホストがWindowsの場合、転送モードの設定を”ASCII”にすることで適切な改行コードで転送できます。転送手順につきましては、Webページをご参照ください。[2]

コンパイル

並列コンピュータ上でコンパイルします。

●Fortran （自動並列・OpenMP）

【形式】 **sxf90 オプション ソースファイル名**

主なオプション

-Pauto	自動並列化機能を利用する。
-Popenmp	OpenMP を利用する。
-pi	インライン展開を行う。
-R5	ベクトル化／並列化状況を表示した編集リストの出力
-ftrace	手続きごとの性能情報の取得

ソースファイル名

Fortran のソースプログラムファイル名を指定します。複数のファイルを指定するときは、空白で区切ります。
ソースファイル名には、サフィックス.f90か.F90（自由形式）、または.fか.F（固定形式）が必要です。

●Fortran （MPI）

【形式】 **sxmpif90 オプション MPI ソースファイル名**

主なオプション

-pi	インライン展開を行う。
-R5	ベクトル化／並列化状況を表示した編集リストの出力
-ftrace	手続きごとの性能情報の取得

ソースファイル名

[2] http://www.cc.tohoku.ac.jp/service/USE/index.html「システム利用の手引き」

Fortran のソースプログラムファイル名を指定します。複数のファイルを指定するときは、空白で区切ります。
ソースファイル名には、サフィックス.f90か.F90(自由形式)、または.fか.F(固定形式)が必要です。

●C/C++ （自動並列・OpenMP）

【形式】 **sxcc オプション ソースファイル名**

主なオプション

-Pauto	自動並列化機能を利用する。
-Popenmp	OpenMP を利用する。
-pi	インライン展開を行う。
-ftrace	手続きごとの性能情報の取得

ソースファイル名

C/C++のソースプログラムファイル名を指定します。複数のファイルを指定するときは、空白で区切ります。
ソースファイル名にはサフィックス .c 、C++プログラムのソースファイル名にはサフィックス .cc または .C が必要です。

●C/C++ （MPI）

【形式】 **sxmpicc オプション ソースファイル名**

主なオプション

-pi	インライン展開を行う。
-ftrace	手続きごとの性能情報の取得

ソースファイル名

C/C++のソースプログラムファイル名を指定します。複数のファイルを指定するときは、空白で区切ります。
ソースファイル名にはサフィックス .c 、C++プログラムのソースファイル名にはサフィックス .cc または .C が必要です。

プログラムの実行

コンパイルして作成された実行形式ファイルを実行するには、以下の2つの処理方法があります。

通常はバッチ処理を利用します。

バッチ処理

バッチ処理は、実行の手続きをジョブという単位でジョブ管理システムに登録し、一括に処理します。ジョブ管理システムは NQSⅡ(Network Queuing SystemⅡ)を用意しており、ジョブの操作は NQSⅡのコマンドで行います。通常のプログラム(長時間実行するプログラム、並列実行するプログラム等)はバッチ処理で実行します。

会話型処理

会話型処理は、コマンドラインでプログラムを実行する形式です。CPU 時間や使用できるメモリサイズに制限がありますので、短時間の演算やデバッグ作業にお使いください。スーパーコンピュータ SX-7 にログインして実行します。

バッチ処理

SX-7,SX-7C のプログラム実行は、並列コンピュータ上で NQSⅡのコマンドを用いて操作します(図2)。

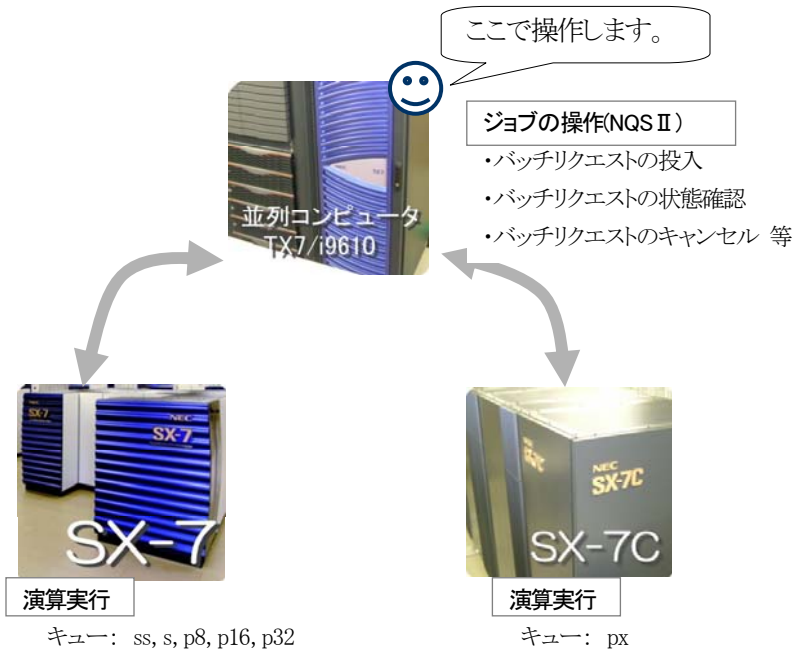

図2 バッチ処理の構成

まず NQSⅡにプログラムの実行を依頼するため、実行の手続きを書いたシェルスクリプトファイルを作成します。作成したファイルはバッチリクエストとして NQSⅡに投入することで、実行の依頼をします。NQSⅡではキューと呼ばれるジョブクラスがあり、並列数やメモリサイズの違いにより複数のキューを設定しています。プログラムに合わせ適切なキューを1つ選択し投入します。リクエストの順番が来ましたら、NQSⅡは自動的に実行します。

リクエスト投入後は、リクエスト状態の確認やキューの込み具合等の確認、また投入済みのリクエストをキャンセルすることも可能です。プログラムが終了するとリクエストはキューの情報から消え、標準出力ファイルと標準エラー出力ファイルが出力されます。

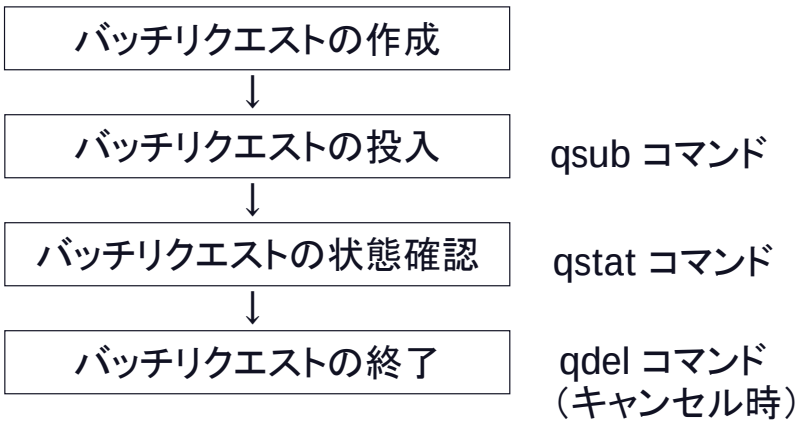

図3 NQSⅡによるリクエストの流れ

バッチリクエストの作成

バッチリクエスト用のシェルスクリプトファイルを作成します。プログラムの実行手続きを、通常のシェルスクリプトと同じ形式で記述します。csh スクリプトと sh スクリプト、どちらでも記述できます（以降、解説は csh スクリプトとします）。リクエストファイル名は任意です。

リスト4はバッチリクエストファイルの一例です。実行形式ファイル a.out を実行する手続きを記述しています。

リスト4 バッチリクエストファイル例

```
# test job-a              コメント行
cd work                   作業ディレクトリへ移動
a.out                     実行形式ファイル名
```

・#が先頭の行は、コメント行です。動作には影響しません。
・cd work で作業ディレクトリ（実行形式ファイルのあるディレクトリ）へ移動します。省略するとホームディレクトリを指定したことになります。
・a.out はコンパイルして作成した実行形式ファイルです。あらかじめ会話型処理で作成しておきます。自動並列、OpenMP 用オブジェクトも、同じ形式で指定します。

●作業ディレクトリの指定

NQSⅡ用の環境変数のひとつとしてPBS_O_WORKDIR 変数があります。この変数には、qsubコマンドを実行した時点のカレントディレクトリが設定されます。

NQSⅡの作業ディレクトリは規定値でホームディレクトリとなりますので、通常 cd コマンドで実行ファイルのある作業ディレクトリに移動する必要があります。 PBS_O_WORKDIR 変数を設定すること

で、ディレクトリの具体名を記述する必要がなくなります。

リスト5　バッチリクエストファイル（環境変数 PBS_O_WORKDIR の指定）

```
# test job-a1
cd $PBS_O_WORKDIR                作業ディレクトリを環境変数で指定
a.out
```

●実行時のデータファイル指定

Fortran で、入出力ファイルを割り当てる環境変数 F_FF*nn* です。

リスト6　バッチリクエストファイル（入出力ファイルの指定例）

```
# test job-b
setenv F_FF11 datafile          装置番号 11 に、ファイル datafile を割り当てる
cd $PBS_O_WORKDIR
a.out < infile > outfile        標準入出力ファイルはリダイレクションでも可能
```

バッチリクエストの投入

プログラムの実行は、作成したバッチリクエストを NQSⅡに投入することで行います。投入されたリクエストは、順番が来ると自動的に実行されます。

【形式】　**qsub　オプション　バッチリクエストファイル名**

システムからのメッセージがリスト7の形式であれば、リクエストは正常に受け付けられています。リクエスト ID(1234)は一意なもので、リクエストの状況確認やキャンセル等、ジョブの操作の際に必要となります。

リスト7　qsub コマンドの実行例

```
gen$ qsub –q p8 job-a
Request 1234.job submitted to queue: p8.
```

主な qsub コマンドオプション

-q	リクエストを投入するキュー名（クラス名）を指定します。（必須）
-N	リクエスト名（ジョブ名）を指定します。指定がなければ、バッチリクエストファイル名がリクエスト名になります。
-o	標準出力のファイル名を指定します。指定がなければ、リクエスト投入時のディレクトリに「リクエスト名.o リクエスト ID」のファイル名で出力されます。

-e	標準エラー出力のファイル名を指定します。指定がなければ、リクエスト投入時のディレクトリに「リクエスト名.e リクエスト ID」のファイル名で出力されます。
-j o	標準エラー出力を標準出力と同じファイルへ出力します。
-l cputim_job=hh:mm:ss	実行打ち切りの CPU 時間を指定します。設定時間は、時:分:秒を hh:mm:ss の形式で指定します。この指定がなければ無制限となります。並列処理で実行するときは、各プロセスの合計 CPU 時間を指定します。
-m b	リクエストの実行が開始したときにメールが送られます。
-m e	リクエストの実行が終了したときにメールが送られます。
-M メールアドレス	リクエストに関するメールの送信先を指定します。指定がなければ、「利用者番号@gen.isc.tohoku.ac.jp」宛に送られます。

●キュー

-qオプションで指定するキュー名の一覧です。単一CPUで実行するプログラムはssかs、並列プログラムは p8,p16,p32,px のいずれかに投入します。並列数に応じて適切なキューにリクエストを投入してください。並列用のキュー(p8,p16,p32,px)を利用するには、並列用のオブジェクトを作成している必要があります。

表2 キュー名

キュー名 （ジョブクラス）	利用可能 CPU 数 （並列数）	CPU 時間	メモリサイズ制限 （GBytes）
ss	4	1 時間	8
s	4	無制限	8
p8	8	無制限	64
p16	16	無制限	128
p32	32	無制限	256
px(MPI 専用)	40	無制限	ノードあたり 128 合計 640

●qsub コマンドオプションの埋め込み

qsub コマンドに毎回オプションを入力することもできますが、手間を省くためバッチリクエストファイルに指定しておくこともできます。

指定方法は、最初のシェルコマンドより前の行に、#PBS という文字列を先頭に指定します。#PBS の後に空白を一文字以上入れ、指定したいオプションを続けます。一行に複数のオプション指定も可能です。リスト8は、2行目で p8 クラスのキューを指定(-q)、3行目で標準エラー出力を標準出力ファイルにひとまとめにする(-jo)、さらにリクエスト名を reqname とする(-N)を、それぞれ指定し

ています。
　またコマンド列と埋め込みオプションに、同じオプションを指定した場合にはコマンド列の方を有効とします。

リスト8　バッチリクエストファイル（オプションの埋め込み）

```
# test job-a2
#PBS –q p8                          埋め込みオプション
#PBS –jo –N reqname                 埋め込みオプション（複数）
cd $PBS_O_WORKDIR
a.out
```

バッチリクエストの状態確認（1）

投入したリクエストの状態を表示します。実行待ち状態のときは、待ち順も表示します。

【形式】　**qstat**

リスト9　qstat コマンド表示例

```
gen$ qstat
 RequestID      ReqName  UserName Queue    Pri STT S   Memory  ACCPU Elapse R H M Jobs
 -------------- -------- -------- -------  --- --- - --------  ----  ------ - - - ----
   353.job       reqname  x2***9  p8        0  RUN  -  732.1B 42350 43012   Y Y Y    1
2:359.job       jobA     x2***9  p32       0  QUE  -    0.0B  0.0     0    Y Y Y    1
5:366.job       jobB     x2***9  p32       0  QUE  -    0.0B  0.0     0    Y Y Y    1
```

主な表示項目の解説

RequestID	リクエスト ID 待ち状態(QUE)のリクエストについては、先頭に待ち順の番号がつきます。番号がないのは、実行中（RUN）です。
ReqName	リクエスト名
UserName	ジョブの所有者
Queue	キュー名
STT	ステータス （QUE：待ち、RUN：実行中）
Memory	使用メモリサイズ （Byte）
ACCPU	演算時間(sec)／並列演算の場合、使用 CPU の総演算時間 （sec）
Elapse	経過時間 （sec）

システム内に自分のジョブが存在しない場合は、ヘッダのみ表示されます。

リスト10 ジョブの終了

```
gen$ qstat
RequestID      ReqName  UserName Queue    Pri STT S   Memory  ACCPU  Elapse R H M Jobs
-------------- -------- -------- ------- --- --- - -------- ---- ------ - - - ----
```

ヘッダのみ表示されます

バッチリクエストの状態確認(2)

キューの情報を表示します。各ジョブクラスの件数が表示されますので、サーバの混雑度がわかります。

【形式】 **qstat -Q**

リスト11 -Q オプションの表示例

```
gen$ qstat -Q
[EXECUTION QUEUE] Batch Server Host: job
====================================
QueueName       SCH JSVs ENA STS  PRI  TOT ARR WAI QUE PRR RUN POR EXT HLD HOL RST SUS MIG STG CHK
--------------- --- ---- ------- ---- ---------------------------------------------------------
a16               0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
a32               0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
a64               0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
a8                0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
am                0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
as                0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
p16               1    4 ENA ACT   32    1   0   0   0   0   1   0   0   0   0   0   0   0   0
p32               1    4 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
p8                1    2 ENA ACT   32    4   0   0   2   0   2   0   0   0   0   0   0   0   0
px                1    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
s                 0    1 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
ss                0    1 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0
--------------- --- ---- ------- ---- ---------------------------------------------------------
 <TOTAL>                            5   0   0   2   0   3   0   0   0   0   0   0   0   0   0
--------------- --- ---- ------- ---- ---------------------------------------------------------
```

枠で囲った箇所が、スーパーコンピュータ SX-7,SX-7C のキューです

主な表示項目の解説

QueueName	キュー名
TOT	リクエストの総数
QUE	待ちの件数
RUN	実行中の件数

バッチリクエストのキャンセル

投入済みのリクエストを取り消すこともできます。

【形式】　**qdel** リクエスト ID

リスト12 qdel コマンドの表示例

```
gen$ qdel 1234
Request 1234.job was deleted.
```

会話型処理

会話型処理は、短時間の演算やデバッグ作業に使用します。一般的な UNIX を利用する手順と同様で、スーパーコンピュータ SX-7 にログインしコマンドラインから実行形式ファイル名を入力し実行する形式です(リスト13)。表3は会話型処理の制限値です。

ホスト名

スーパーコンピュータ SX-7	**super.isc.tohoku.ac.jp**

リスト13 会話型処理の例(a.out を実行する)

```
yourhost$ ssh super.isc.tohoku.ac.jp -l 利用者番号      superにログインする
    :
super$ a.out
  (実行中)
super$
```

表３　会話型処理の制限値

CPU 時間(時間) (並列処理は総 CPU 時間)	メモリサイズ (GBytes)	同時使用 CPU 台数 (並列数)
1	8	4

また、SX-7C で会話型処理の実行はできませんので SX-7 上で行います。この場合、SX-7C 専用オプションでコンパイルしたオブジェクトファイルは、SX-7 で実行できませんのでご注意ください。

px クラスで MPI プログラムを実行する

px クラスは、40 並列の MPI プログラム専用キューです。

コンパイル

px クラスでは、MPI に自動並列または OpenMP を組み合わせて並列処理を実行することも可能です。その場合は、コンパイル時に自動並列オプション-Pauto または-Popenmp を付けコンパイルします。

【形式】 sxmpif90 オプション MPI ソースファイル名

主なオプション

-Pauto	自動並列化機能を利用する。
-Popenmp	OpenMP を利用する。
-sx8	SX-7C,SX-8 向けの命令を生成する。
-pi	インライン展開を行う。
-R5	ベクトル化／並列化状況を表示した編集リストの出力
-ftrace	手続きごとの性能情報の取得

ソースファイル名

Fortran のソースプログラムファイル名を指定します。複数のファイルを指定するときは、空白で区切ります。
ソースファイル名には、サフィックス.f90 か.F90(自由形式)、または.f か.F(固定形式)が必要です。

バッチリクエスト作成 MPI で 40 並列

複数ノードを利用するため、mpirunコマンドに-nn オプションも必要となります。px クラスの場合は 5 ノード使用しますので-nn 5 と指定します。

【形式】 mpirun -np 総プロセス数 -nn 使用ノード数 実行形式ファイル名

リスト14は MPI プログラムを 40 並列実行する例です。-np に並列数(総プロセス数)である 40、-nn には 40 並列によって使用するノード数 5 を、それぞれ指定します。

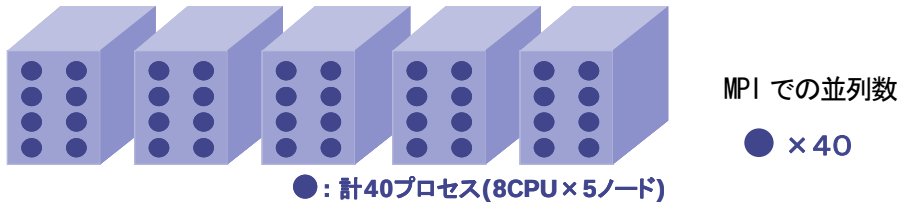

図4 MPI で 40 並列

リスト14　バッチリクエストファイル（MPI で 40 並列実行）

```
# test job-M40
#PBS –q px
#PBS –jo –N reqname
cd $PBS_O_WORKDIR
mpirun –np 40 –nn 5 a.out      -np は総プロセス数、-nn は使用ノード数
```

バッチリクエスト作成
MPIと自動並列(またはOpenMP)の組み合わせ

並列の組み合わせ数は、MPI で 5 並列(プロセス)と自動並列で 8 並列(タスク)の 5×8=40 並列とします。リスト15に示すように、-np にMPI の並列数である5、-nn には使用するノード数5を指定します。実行形式ファイル a.out は　-Pauto や-Popenmp でノード内並列化オプションを付けコンパイルしておくことが条件です。ノード内の並列数 8 は指定する必要はありません。

図5　MPI5並列と自動並列8並列の組み合わせ（ハイブリッド並列）

リスト15　バッチリクエストファイル（ハイブリッドで 40 並列実行）

```
# test job-H40
#PBS –q px
#PBS –jo –N reqname
cd $PBS_O_WORKDIR
mpirun –np 5 –nn 5 a.out       -np は総プロセス数、-nn は使用ノード数。
```

px クラスで環境変数を設定している場合

px クラスで環境変数を設定しているときは、MPIEXPORT 環境変数により各ノードにも設定を反映させる必要があります。この設定がないと1つのノードにしか環境変数が反映されませんのでご注意ください。

リスト16 環境変数 MPIEXPORT の記述例

```
# test job-M40B
#PBS –q px
#PBS –jo –N reqname
setenv F_FF11 datafile
setenv F_FF12 datafile-B
setenv F_UFMTENDIAN  30,40
setenv MPIEXPORT “F_FF11 F_F12 F_UFMTENDIAN”
                                        “ ”内に、設定する環境変数を記述する
                                                  複数の場合、空白で区切る
cd $PBS_O_WORKDIR
mpirun –np 40 –nn 5 a.out
```

MPI 用実行性能情報の表示

MPIPROGINF 環境変数の設定により、プログラム性能情報の表示形式を変更することができます。MPIPROGINF の値と表示内容は以下のとおりです。

DETAIL	性能情報を集約形式で出力します
ALL_DETAIL	性能情報を拡張形式で出力します。全ランクの情報も出力します。

性能情報はプログラム実行終了後、標準エラー出力ファイルに出力されます。

リスト17 バッチリクエストファイル（MPIPROGINF の指定）

```
# test job-M40C
#PBS –q px
#PBS –jo –N reqname
setenv MPIPROGINF DETAIL          実行性能情報表示の指定
cd $PBS_O_WORKDIR
mpirun –np 40 –nn 5 a.out
```

その他

プログラムの使用メモリサイズ

プログラムの使用するメモリサイズを size コマンドで知ることができます。なお、allocate 等で動的に確保するメモリサイズは含まれません。コマンドの実行は、super 上で行います。

【形式】 **size** 実行形式ファイル名
【形式】 **size –fl** [並列数] 実行形式ファイル名　　（自動並列または OpenMP 用）

リスト18 使用メモリサイズの表示

```
super$ size a.out
1046912 + 140272 + 418928 = 1606112          1,606,112 バイト使用します
```

自動並列プログラムの場合は、-fl オプションの後に並列数を指定します。

リスト19 使用メモリサイズの表示（自動並列用実行ファイル、16 並列の例）

```
super$ size –fl 16 a.out
1046912(.text) + 140272(.data) + 418928(.bss) + 181855(.comment) +
4496(.whoami) + 1048576(logical task region) * 16 = 18569679
                                        16 並列実行で、18,569,679 バイト使用します
```

バイナリファイルの扱い（Fortran の場合）

他のサーバで作成したバイナリファイルを扱う場合、注意が必要です。スーパーコンピュータ SX-7 および SX-7C は Big-Endian 仕様ですので、Little-Endian 仕様のバイナリファイルを扱うには、環境変数 F_UFMTENDIAN で Little から Big に変換処理を行います。

Little-Endian 仕様のファイルを扱う設定（csh 形式）

【形式】 **setenv F_UFMTENDIAN u[,u]...** （**u** は Little-Endian ファイルの装置番号）

リスト20 バッチリクエストファイル例（Little-Endian の変換設定）

```
# test job-endian
#PBS –q p8
#PBS –jo –N reqname
setenv F_UFMTENDIAN 30,40       装置番号 30,40 のファイルは Little-Endain
cd $PBS_O_WORKDIR
a.out
```

マニュアル

センター1階利用相談室に設置しているマニュアルです。

FORTRAN90/SX プログラミングの手引
FORTRAN90/SX 言語説明書
FORTRAN90/SX 並列処理機能利用の手引
C++/SX プログラミングの手引
MPI/SX 利用の手引

科学技術計算ライブラリ ASL/SX 利用の手引(基本機能編 1/4)
科学技術計算ライブラリ ASL/SX 利用の手引(基本機能編 2/4)
科学技術計算ライブラリ ASL/SX 利用の手引(基本機能編 3/4)
科学技術計算ライブラリ ASL/SX 利用の手引(基本機能編 4/4)
科学技術計算ライブラリ ASL/SX 利用の手引(高速機能編)
科学技術計算ライブラリ ASL/SX 利用の手引(並列処理機能編)

おわりに

ジョブ管理システム NQSⅡの使い方、MPI プログラムの利用手順等、これまでの利用方法との変更点を中心に解説しました。研究の強力なツールとしてセンターのシステムをご活用いただければ幸いです。ご不明な点、ご質問等ございましたら、お気軽にシステム管理までお問い合わせください。