スーパーコンピュータ SX-７の利用法

システム管理掛
システム運用掛
スーパーコンピューティング研究部

１．　はじめに

情報シナジーセンターの大規模科学計算システムは、スーパーコンピュータ、並列コンピュータ、およびファイルサーバより構成されています（図１）。

スーパーコンピュータ（SX-７、日本電気㈱製）は、Fortran の Do ループや C／C++ の for ループを高速に実行することのできる、ベクトル型スーパーコンピュータです。並列コンピュータ（TX7/AzusA、日本電気㈱製）は、64bit プロセッサ Itanium™（IA-64）を CPU にもち、１つのプログラムを複数の CPU で同時に実行する並列処理によって、高速化を図ります。ファイルサーバ（NX7000、日本電気㈱製）は、スーパーコンピュータと並列コンピュータに共通のファイル装置で、プログラムやデータのファイルは両方のシステムからアクセスできますので、ベクトル化率の高いプログラムはスーパーコンピュータ、ベクトル化率の低いものは並列コンピュータ、というようにプログラムに応じて使い分けることができます。

以下では、スーパーコンピュータの利用法について説明します。

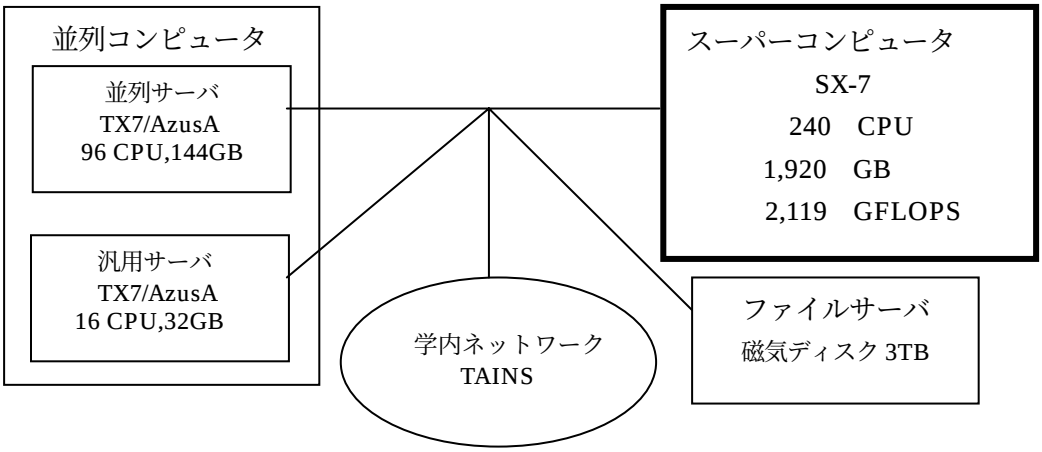

図１　システム構成

２．　スーパーコンピュータ SX-7

スーパーコンピュータ SX-7 は、Fortran の Do ループや C／C++の for ループを、高速のベクトル命令を用いて実行する（ベクトル化という）ベクトル型のスーパーコンピュータです。SX-7 は 8.83GFLOPS のベクトル演算性能をもつ 32 個の CPU で１つのノードを構成しますので、１つのノードのベクトル演算性能は 8.83×32＝282.56GFLOPS となり、極めて高速です。また、ベンチマークプログラムとして世界的に用いられている LINPACK HPC においては、SX-7 の１つのノードの演算性能は 273.7GFLOPS に達し、非常に高い実効性能を誇っています。さらに、１つのノードは 256GB という極めて大容量のメモリを備えていますので､演算性能においてもメモリ容量においても、他では実行できないような大規模の計算が可能です。

SX-7 の Fortran／C／C++のコンパイラは、プログラムを高速に実行するために、上述のベクトル化の機能に加えて、並列処理の機能も有しています。特に、自動並列化機能を利用すると、プログラムを書き換えることなしに並列処理が可能となります（コンパイル時に、自動並列化のオプションを指定します)。後述のジョブクラス表（表３）の p32 というキューで実行する場合は､１つのプログラムで 32 個の CPU を使用します(すなわち、ノードのすべての CPU と 256GB のメモリを占有します)。なお、並列処理としては、この自動並列化機能のほかに、HPF（Fortran のみ)、OpenMP、MPI も利用できます。

３．　会話型処理

プログラムおよびデータの作成や編集は並列コンピュータにログインして行います。また、Secure　Shell（OpenSSH）や emacs 等の各種フリーソフトの利用、Fortran や C/C++プログラムのコンパイルも並列コンピュータで行います。プログラムの実行は、バッチ処理での実行が主で、そのためスーパーコンピュータへのジョブの投入は並列コンピュータで行います（「４．バッチ処理」参照)。なお、直接スーパーコンピュータにログインして実行する方法（「3.5 会話型処理でのプログラム実行」参照）もあります。

3.1 ログイン

センターに利用登録した時点でスーパーコンピュータ、並列コンピュータ、ファイルサーバが利用可能となります。ログイン名とパスワードは全サーバで共通です。ログイン名は利用者番号です。なお、パスワードは機密保護のためときどき yppasswd コマンドで変更してください。

並列コンピュータへのログインには、Secure Shell（SSH）を利用します（telnet でも接続できますが、機密保護のためにできるだけ SSH を使うことをお奨めします)。ホスト名はつぎのとおりです。

gen.cc.tohoku.ac.jp

利用者番号（ログイン名)、パスワードを順次入力します。ホームディレクトリは「/uhome/利用者番号」です。利用登録時のログインシェルとして csh を設定していますので、tcsh に変更する場合は ypchsh コマンドをご利用ください。

[例] 並列コンピュータへのログイン

```
% ssh gen.cc.tohoku.ac.jp –l 利用者番号
Enter passphrase ・・・・・・ : パスワード
%
```

3.2 プログラム言語

プログラム言語として表１に示すものがあり、表のコマンド欄で示すコマンド名で利用します。各コマンドのオプションについては、表 2 に示すマニュアルおよび man コマンドで知ることができます。マニュアルは利用相談室で閲覧できます。

表 1 プログラム言語とコマンド

言　語	コマンド名	備　　考
Fortran	sxf90	ISO/IEC 1539-1:1997 Fortran 準拠 （自動並列機能、OpenMP（Version1.1)）
MPI/Fortran	sxmpif90	MPI　forum　MPI2.0 準拠
HPF	sxhpf	HPF2.0　準拠
C	sxcc	ISO/IEC 9899:1990 C 準拠 （自動並列機能、OpenMP（Version1.0)）
C++	sxc++	ISO/IEC 14882:1998 C++準拠 （自動並列機能、OpenMP（Version1.0)）
MPI/C	sxmpicc	MPI　forum　MPI2.0 準拠
MPI/C++	sxmpic++	MPI　forum　MPI2.0 準拠

表2　マニュアル

言　語	マニュアル
Fortran	FORTRAN90/SX 言語説明書（日本電気） FORTRAN90/SX プログラミングの手引き（日本電気）
HPF	HPF/SX　V2 利用の手引（日本電気）
C	Ｃプログラミングの手引き（日本電気）
C++	Ｃ＋＋言語説明書（日本電気） Ｃ＋＋プログラミングの手引き（日本電気）
MPI	MPI/SX 利用の手引（日本電気）
OpenMP	http://www.openmp.org/を参照

3.3 Fortran プログラム

3.3.1 コンパイル

Fortran のソースプログラムは sxf90 コマンドでコンパイル、リンクをします。数値計算ライブラリ ASL/SX（日本電気㈱）が利用できます（ライブラリ指定オプション –l は不要です）。

[形式] **sxf90** *options source*

options ：コンパイルのオプションを指定します。man コマンドあるいは「FORTRAN90/SX プログラミングの手引き」にオプションの種類と既定値を示してあります。変更する時は、大文字/小文字の区別を注意してください。なお、リンクが失敗したとき実行形式オブジェクトプログラムは作成されません。以下に代表的なオプションを記します。

-Pauto	自動並列化機能の利用
-Popenmp	**OpenMP** の利用
-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

source ： 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。

・Fortran のソースプログラムファイル名にはサフィックス「.f」「.f90」または「.F」「.F90」が必要です(注)。

・ソースプログラムの記述形式に応じ、つぎのオプションを指定します。

-f0 Fortran の固定形式

-f4 Fortran90 の自由形式

(注)サフィックス「.f、.f90、.F、.F90、.c、.s」以外のファイルは、オブジェクトプログラムと解釈されリンク時の入力となります。

[例] 自動並列化機能を用いたコンパイル

```
sxf90  -P auto  source.f
```

3.3.2 実行時のデータファイル指定

Fortran プログラム実行時にデータファイルを使用する場合、setenv コマンドでファイル名を指定します。この指定はプログラムの実行前に行います。

[形式] **setenv F_FF***nn* ファイル名

nn は装置番号で、1～9 までは 01～09 と指定します。

また、装置番号 0、5、6 はリダイレクションで割り当てることもできます。

a.out < infile	装置番号 5 のデータを infile から入力する
a.out > outfile	装置番号 6 のデータを outfile に出力する
a.out >& outfile	装置番号 0 と 6 のデータを outfile に出力する

3.4 C/C++ プログラム

3.4.1 コンパイル

C／C++のソースプログラムはそれぞれ sxcc、sxc++コマンドでコンパイル、リンクをします。数値計算ライブラリ ASLCINT/SX（日本電気㈱）が利用できます（ライブラリ指定オプション –l は不要です）。

[形式] **sxcc** *options source*
sxc++ *options source*

options : コンパイルのオプションを指定します。man コマンドあるいは「C プログラミングの手引き」にオプションの種類と既定値を示してあります。変更する時は、大文字/小文字の区別を注意してください。なお、リンクが失敗したとき実行形式オブジェクトプログラムは作成されません。以下に代表的なオプションを記します。

-Pauto	自動並列化機能の利用
-Popenmp	**OpenMP** の利用
-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

source : 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。C のソースプログラムファイル名にはサフィックス「.c」、C++のソースプログラムファイル名にはサフィックス「.cc」または「.C」が必要です。

[例] 自動並列化機能を用いたコンパイル

```
sxcc  -P auto  source.c
```

3.5 会話型処理でのプログラム実行

実行形式プログラム（a.out）を会話型処理で実行する場合は、スーパーコンピュータに telnet コマンドでログインしてください。ホスト名は **super.cc.tohoku.ac.jp** です。利用者番号（ログイン名）、パスワードを順次入力します。なお、会話型処理でのプログラム実行は、CPU 時間 1 時間、メモリサイズ 8GB に制限されていますので、短時間の計算やデバッグ目的にのみお使いください。

[例] スーパーコンピュータへのログイン

```
% telnet super.cc.tohoku.ac.jp
Connected to super.cc.tohoku.ac.jp
                .
login: 利用者番号
password:パスワード
%a.out
```

４． バッチ処理

バッチ処理では、長時間のプログラムを実行することができます。また、最大 256GB という大規模メモリを使用することもできます。プログラムを実行するためには、ジョブファイルの作成、ジョブの投入という手順を踏みます（会話型処理での実行は、3.5 を参照ください)。

4.1 ジョブファイルの作成

プログラムを実行するためには、一連のコマンドを記述したシェルスクリプトファイル（ジョブファイル）を作成します。図２はジョブファイルの例で、先頭の「#!/bin/csh -x」はプログラムを実行するときのシェルの指定で、ここでは csh を指定しています。つぎの「#」で始まる行はコメントとして扱われます。「cd dir1」はプログラムを実行するためのディレクトリへの移動です（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)。続く「setenv」は Fortran プログラムを実行するときのデータファイルの指定です（３.３.２参照)。最後の「a.out」は実行すべき実行形式オブジェクトファイルの指定です。なお、Fortran／C／C++のコンパイルはジョブの中では行えません。あらかじめ会話型処理でコンパイルし実行形式オブジェクトファイルを作成しておいてください。

```
%cat  job1
#!/bin/csh -x      …実行シェルの指定（省略時は sh）
#      job 1
#
cd  dir1           …プログラムを実行するためのディレクトリ(dir1)に移動
setenv  F_FF01  datafile     …データファイルの指定
setenv  F_FF02  outfile              〃
a.out                        …プログラムの実行
```

図２　ジョブファイルの例

4.2 ジョブの投入

スーパーコンピュータにジョブを投入するには、qsub コマンドを用います。つぎの［例１］は qsub コマンドの指定例で、ジョブを実行するキュー名（-q p32)、リクエスト名（-r program1)、およびジョブファイルを指定しています。qsub コマンドが受け付け

られると、つぎのようにリクエスト ID を含むメッセージが表示されます。ここで、「123.super」がリクエスト ID で、後にジョブの状況表示や取り消しをするときに必要となります。また、「123」の部分はリクエスト番号といい、実行結果（標準出力、標準エラー出力）のファイル名の一部に使われます。

[例１] ジョブの投入（qsub コマンド）

```
%qsub -q p32 -r program1 job1                    ジョブ投入
 Request 123.super submitted to queue : p32      123.super はリクエスト
```

[形式] **qsub -q** キュー名　オプション　ジョブファイル名

・キュー名 ： ジョブを実行するキュー名を指定します（表 3）。p8、p16、p32 は並列処理用のキュー、ss と s は非並列用のキューです。

表 3　ジョブクラス

キュー名	利用 CPU 数	CPU 時間	メモリ制限
ss	**1**	1 時間	**8GB**
s	**1**	無制限	**8GB**
p8	**8**	無制限	**64GB**
p16	**16**	無制限	**128GB**
p32	**32**	無制限	**256GB**

・オプション

-r request-name：リクエスト名（ジョブ名）を指定します。指定がなければ、コマンド行であたえられたジョブファイル(先頭のパス名を除く）の名前が使われます。

-o outfile： 標準出力をファイル outfile へ出力します。この指定がなければ、ジョブ投入時のカレントディレクトリ下に「リクエスト名．o リクエスト番号」のファイルが作られ出力されます。（ホームディレクトリ下ではないのでご注意ください)。

-e errfile： 標準エラー出力をファイル errfile へ出力します。この指定がなければ、ジョブ投入時のカレントディレクトリ下に「リクエスト名．e リクエスト番号」のファイルが作られ出力されます。（ホームディレクトリ下ではないのでご注意ください)。

-eo ： 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。

-lT ： 打ち切り CPU 時間を指定します。時間は時、分、秒を hh:mm:ss の形式で表します。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。

・ジョブファイル名：　実行するジョブファイル名（スクリプトファイル）を指定します。

［備考］オプションの埋め込み

qsub コマンドのオプションは、ジョブファイルに埋め込むことができます。埋め込みオプションは#@$に続けて指定し、複数行にわたる場合は、#@$行を続けます。#@$のみの行はオプションの最後を示します。例 1 の qsub コマンドのオプションとキュー名の指定をジョブファイルに埋め込んだ例が図 3 です。このとき、qsub コマンドはつぎのように指定します。

[例２]　qsub コマンドの例（オプションの埋め込み）

```
%qsub  job1
Request  234.super submitted to queue : p32

%cat  job1
#
#     job  1
#
#@$-q p32
#@$-r program1
#@$
cd  dir
setenv  F_FF01  datafile
setenv  F_FF02  outfile
a.out
```

図３　オプション埋め込みの例

埋め込みオプションに加えて、qsub コマンド入力時にオプションを追加指定することができます。同じオプションを指定した場合は、コマンドで指定したオプションが優先されます。つぎの例は、標準エラー出力と標準出力を同じファイルに出力するためのオプションを指定するとともに、リクエスト名を program2 に変えています。

[例３]　qsub コマンドの例（オプションの追加）

```
%qsub  -eo   -o ファイル名  -r  program2
Request  345.super submitted to queue : p32
```

4.3 ジョブの取り消し

指定したリクエスト ID を取り消します。

[形式] **qdel [-k]** リクエスト ID

-k : ジョブの状態にかかわらずリクエストを取り消します
-k を省略すると、実行待ちのジョブだけを取り消します

リクエスト ID : 取り消したいリクエスト ID を指定します。

[例] **%qdel -k 234.super**
request 234.super is running ,and has been signaled.

4.4 ジョブの状況表示

4.4.1 qstatr コマンド

投入したジョブの状態を表示します。つぎの例では、リクエスト ID123.super はジョブクラス P32 で実行中であり、使用メモリ 100GB（④MEMORY）、演算時間は 32915 分（⑤TIME）であることを示しています。また、345.super で WAT すなわち待ち状態（⑥STT）です。

```
% qstatr
①               ②               ③               ④      ⑤     ⑥
REQUEST ID      NAME     OWNER    QUEUE    PRI NICE MEMORY  TIME  STT  JID    R
--------------- -------- -------- -------- ---- --- ------ ------ --- ------ -
123.super       program1 a20000   P32        31   0   100G  32915m RUN   645 -
345.super       program2 a20000   p32        31   0                WAT       -
--------------------------------------------------------------------------------
```

①リクエスト ID
②リクエスト名
③ジョブクラス名
④使用メモリ
⑤使用演算時間
⑥ジョブの状態（WAT は実行待ち、RUN は実行中）

4.4.2 qstatq コマンド

各ジョブクラスの状況を表示します。つぎの例では、ジョブクラス p32 は全部で 13 件（②TOT）のジョブがあり、実行中が 5 件（③RUN）、実行待ちが 5 件（④WAI）であることを示しています。

```
% qstatq
①                                                ②       ③  ④
QUEUE NAME       ENA STS PRI/BPR/ TMS /MPR RLM  TOT QUE RUN WAI HLD SUS ARR EXT
-------------- ------------------------------ ------------------------------
ss               ENA RUN  31/400/10000/-11   2    2   0   2   0   0   0   0   0
s                ENA RUN  31/400/10000/-11   8    8   0   8   0   0   0   0   0
p8               ENA RUN  31/400/10000/-11   3    3   0   3   0   0   0   0   0
p16              ENA RUN  31/400/10000/-11   2    2   0   2   0   0   0   0   0
p32              ENA RUN  31/400/10000/-11   5   13   0   5   8   0   0   0   0
-------------- ------------------------------ ------------------------------
   <TOTAL>                                   50   28   0  20   8   0   0   0   0
-------------- ------------------------------ ------------------------------
```

①ジョブクラス
②ジョブの総数
③実行中のジョブ数
④実行待ちのジョブ数

4.4.3 qstat コマンド

各ジョブクラスのジョブの状態を表示します。つぎの例では、リクエスト名 job3、job4 は実行待ちで job3 は 2 番目（2:)、job４は 8 番目（8:）です。

```
% qstat
ss@super;  type=PIPE;  [ENABLED, INACTIVE];  pri=35
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;
s@super;  type=PIPE;  [ENABLED, INACTIVE];  pri=35
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;
p8@super;  type=PIPE;  [ENABLED, INACTIVE];  pri=35
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;
p16@super;  type=PIPE;  [ENABLED, INACTIVE];  pri=35
  0 depart;  0 route;  0 queued;  0 wait;  0 hold;  0 arrive;

①  ②          ③
p32@super;  type=PIPE;  [ENABLED, RUNNING];  pri=31
                                    ④
  0 depart;  1 route;  0 queued;  8 wait;  0 hold;  0 arrive;
         ⑤                   ⑥                    ⑦
         REQUEST NAME         REQUEST ID           USER  PRI    STATE      PGRP
<1 request WAITING>
    2:          job3      356.super             a20000  31    WAITING
<5 requests WAITING>
    8:          job4      360.super             a20000  31    WAITING
```

①ジョブクラス
②ホスト名
③実行待ちのキュー（type=PIPE）
④実行待ちのジョブ数
⑤リクエスト名
⑥リクエスト ID
⑦利用者番号

５．おわりに

スーパーコンピュータ SX-7 は、科学技術計算や各種シミュレーションで多用されるベクトル演算、行列演算を高速に実行するベクトル型のスーパーコンピュータです。SX-7 の Fortran／C／C++は自動並列化機能を有していますので、プログラムを書き換えることなく並列化のオプションを指定してコンパイルするだけで並列処理を行うことができ、SX-7 の高い性能を利用できますので、性能やメモリの制約によっていままではできなかった問題の解明に是非役立てていただきたいと思います。