スーパーコンピュータシステムSX-9 利用ガイド

情報基盤課 システム管理係
サイバーサイエンスセンター スーパーコンピューティング研究部

1章　はじめに

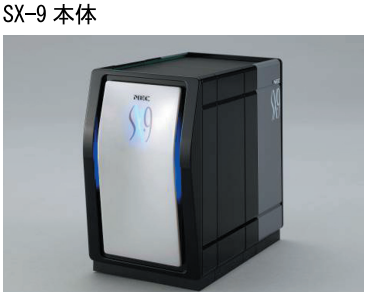
2008 年 3 月、本センターはスーパーコンピュータシステム SX-9 を導入し、新たなシステム構成で運用を開始しています。本稿は、SX-9 システムでのプログラミング利用ガイドとして、プログラムの作成からコンパイル、実行等の使い方をご紹介します。

2章　システムの特徴

ハードウェア

スーパーコンピュータシステムSX-9（日本電気㈱製）は、前システムSX-7 と同じくベクトル並列型スーパーコンピュータを継承しています。1 ノードあたり 16CPU構成で、1CPUあたりの演算性能は 100G FLOPS[1]超と大幅に性能向上しています。物理メモリはノードあたり 1Tバイトの共有メモリ型、バンド幅は1CPUあたり 256Gバイト/秒のデータ転送能力を持ち、大規模かつ高性能な処理を可能にします。本センターでは、全 16 ノードのシステム構成でSX-9 を運用します。

●図1　SX-9 本体



●図2　本センターの SX-9 システム

[1] Floating point number Operations Per Second（浮動少数点演算性能）1秒あたり 100 ギガ回、つまり 1000 億回を超える浮動少数点演算が可能。

●ハードウェアの特徴

世界最速[2]ベクトルプロセッサ
１CPU あたり１０２．４G FLOPS

大規模な共有メモリ型並列処理
１ノードあたり　１６CPU、１Tバイト

圧倒的なデータ転送性能（メモリバンド幅）
１CPU あたり２５６Gバイト/秒

■ベクトル計算機って？

ベクトル演算を行う専用のハードウェアを持つコンピュータです。ベクトル演算は、ループ中で繰り返し処理されるような配列データの演算に対して、逐次的ではなく一括して演算実行するため高速に演算することができます。

ベクトル計算機は科学技術用の数値計算に適しており、大量のデータを繰り返し処理するような大規模計算に向いていると言われています。本センターでは、流体解析や気象解析、電磁界解析をはじめとする大規模シミュレーションにご利用いただいています。

ソフトウェア

UNIX に準拠した SUPER-UX オペレーティングシステムを継承し、ベクトル処理、並列処理に対応したプログラムも従来どおり演算時間に制限無く実行可能です。また、SX-9 に最適化された科学技術用数値演算ライブラリ ASL/SX も引き続きご利用できます。

■プログラミング言語

プログラミング言語は Fortran と C/C++を用意しています。それぞれ自動ベクトル化機能、自動並列化機能を有していますので、既存のプログラムを修正することなくベクトル化、並列化することができます。OpenMP や MPI による並列化プログラムも利用可能です。

■並列化プログラミング

並列化の方法として**表 1** の 3 種類を用意しています。

自動並列化は、単一 CPU で動作する逐次処理プログラムを並列化します。プログラムを改めて書き直

[2] 浮動小数点演算性能(2007 年 10 月時点)

す必要はなく、お持ちのプログラムをそのまま利用することができます。コンパイラが並列化可能な箇所を自動的に判断し並列化します。

OpenMP は、自動並列化と同じく逐次処理プログラムを並列化します。ただ、並列化の判断は自動ではなくユーザが行います。ソース中の並列化したい箇所に並列化指示行を追加します。

MPI は、メッセージ交換用ライブラリによりプロセッサ間通信を行います。データの分割、処理方法等の並列処理手順を明示的に記述した MPI プログラムを作成する必要があります。

一般的に、共有メモリのノード内では自動並列化あるいは OpenMP による並列処理を、分散メモリのノード間では MPI による並列処理という手段を用います。したがって、自動並列化や OpenMP は 16 並列までのノード内並列処理ができます。それ以上の並列数で実行する場合は、複数のノードを必要とするため MPI による並列処理を行います。4 ノード使用する 64 並列と 2 ノード使用する 32 並列は、MPI 並列処理用として用意しています。

自動並列化と OpenMP によるノード内並列処理手順は 4 章で、MPI による並列処理手順は 5 章で、それぞれ紹介します。

●表1　並列化の種類、特徴、並列数

	逐次処理プログラムの並列化	並列化の指示	最大並列数
自動並列化	そのままで可能。	自動（コンパイラ）	16 並列ノード内
OpenMP	指示行を追記することで可能。	手動（ユーザ）	16 並列ノード内
MPI	できない。MPI 用プログラムを作成する。	通信ライブラリを用いプログラミングする。（ユーザ）	64 並列 4 ノード

■互換性

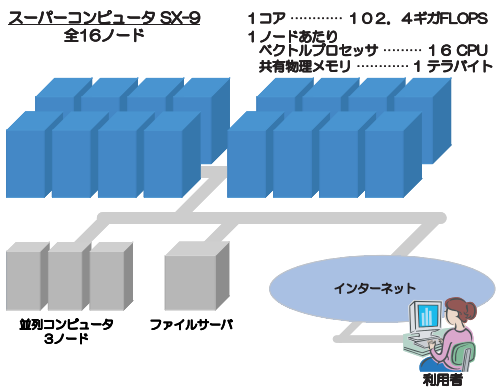
前システム SX-7 とプログラムの互換性はありますが、SX-9 システムで再コンパイルしてから実行してください。SX-9 の性能を十分に引き出すために、最適化機能が強化された SX-9 用コンパイラで再コンパイルします。

3章　システムの構成

本センターでは、SX-9 システム 16 ノードと既設の並列コンピュータシステム（TX7/i9610）の 2 つのシステムをサービスしています。

並列コンピュータシステムは、従来どおり運用し利用方法にも変更はありません。

●図3　システム構成図



4章　プログラミング
～ 逐次処理、ノード内並列処理 ～

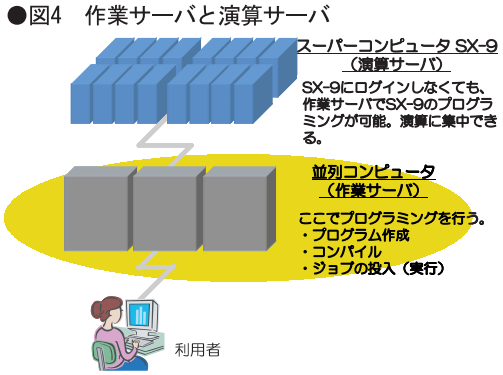
本章では、従来の単一 CPU で実行する逐次処理と、自動並列化および OpenMP によるノード内並列処理について利用手順を紹介します。

MPI による並列化プログラミング手順については、次章で紹介しますが、コンパイルコマンド、ジョブクラス等に違いがある以外は、同じ手順ですのでこの章と合わせてご覧ください。

4-1 フロントエンドサーバ（作業サーバ）

SX-9 用のプログラミングは、既設の並列コンピュータ上で全ての作業を行えるようにしています。ソースファイル作成、コンパイル、実行等一連のプログラミング作業が可能です。SX-9 システムのパフォーマンスをプログラム演算に集中させるために、作業サーバと演算サーバの 2 段構成にしています。

●図4　作業サーバと演算サーバ



4-2 ログイン

作業を行うため並列コンピュータにログインします。リモート接続は、sshコマンドまたはSSH対応リモート接続ソフト[3]をご利用ください。

並列コンピュータの OS は Linux です。ログイン名とパスワードを入力することでログインすることができます。アカウント希望の場合は、受付（庶務係）に利用申請し利用者番号と初期パスワードを発行してもらいます。

並列コンピュータホスト名

```
gen.isc.tohoku.ac.jp
```

●リスト 1　ssh コマンドによる接続例

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

4-3 ホームディレクトリ

ホームディレクトリは、プログラムファイル等を置く自分専用のディスク領域です。ディレクトリ名は、/uhome/利用者番号　です。容量の制限は特に設けていませんが、利用サイズに応じてファイル負担経費が発生します。

なお、ホームディレクトリはスーパーコンピュータシステムと並列コンピュータシステムで共有しています。

ホームディレクトリ

```
/uhome/利用者番号
```

4-4 ソースファイル作成

ソースファイルを作成するためのエディタは、emacs か vi をご利用ください。

研究室等のサーバ、パソコンから転送する場合は、SSH対応のファイル転送ソフト[4]で並列コンピュータに転送してください。この際、ソース（テキスト）ファイルはASCIIモードで転送します。

4-5 コンパイラ

SX-9 用に最適化されたコンパイラ Fortran および C/C++を用意しています。両言語とも並列コンピュータ上でコンパイルします。

前システム SX-7 で利用していたプログラムは、SX-9 システムの性能を十分に引き出すために再コンパイルしてから実行してください。

Fortran コンパイラの仕様

Fortran90/SX（日本電気）	ISO/IEC 1539-1:1997 自動ベクトル化、自動並列化、OpenMP 対応

C/C++コンパイラの仕様

C++/SX（日本電気）	ISO/IEC 14882:1998 自動ベクトル化、自動並列化、OpenMP 対応

4-6 コンパイルする

通常のコンパイルと同様に、それぞれの言語用コマンドでコンパイルします。この項目では単一 CPU で実行する逐次処理と、自動並列化または OpenMP による並列処理、それぞれのコンパイル手順について解説します。

■Fortran プログラムのコンパイル

sxf90 コマンドでコンパイルします。利用したい機能があれば適当なオプションと、Fortran プログラムソースファイル名を指定します。

ソースファイルの拡張子は、自由形式（フリーフォーマット）なら.f90 か.F90、固定形式（7 カラム目から記述）なら.f か.F を付けます（**表 3**）。

並列化コンパイルは、ここで並列数を意識する必要はありません。実行する時点 **4-8-3 ジョブを投入** で希望する並列数を選択することができます。

コンパイル【逐次処理】

```
sxf90 オプション ソースファイル名
```

コンパイル【自動並列化】

```
sxf90 -Pauto オプション ソースファイル名
```

OpenMP プログラムなら、-Pauto の箇所を-Popenmp にします。

表 2 は、主なオプションです。詳細は、man コマンド man sxf90 でご覧ください。

●表2　主なオプション

-Pauto	自動並列化機能を利用する。
-Popenmp	OpenMP 対応機能を利用する。
-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	簡易性能解析機能を利用する
-eC	配列要素参照時、添字の値が範囲内であるか検査する（バウンズチェック）。

[3] Windows であれば、TeraTerm(TTSSH), Putty 等のフリーソフトがあります。

[4] Windows であれば、WinSCP 等のフリーソフトがあります。

●表3　拡張子とフォーマット

拡張子	フォーマット
f90, F90	自由形式
f, F	固定形式

■C プログラムのコンパイル

sxcc コマンドでコンパイルします。利用したい機能があれば適当なオプションと、C プログラムソースファイル名を指定します。

並列化コンパイルは、ここで並列数を意識する必要はありません。実行する時点 **4-8-3 ジョブを投入** で希望する並列数を選択することができます。

コンパイル【逐次処理】

```
sxcc　オプション　ソースファイル名
```

コンパイル【自動並列化】

```
sxcc -Pauto オプション　ソースファイル名
```

OpenMP プログラムなら、-Pauto の箇所を -Popenmp にします。

表 4 は、主なオプションです。詳細は、man コマンド man sxcc でご覧ください。

●表4　主なオプション

-Pauto	自動並列化機能を利用する。
-Popenmp	OpenMP 対応機能を利用する。
-ftrace	簡易性能解析機能を利用する

4-7 プログラムを実行する

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、バッチ処理と会話型処理の 2 通りの方法があります。通常はバッチ処理で実行します。

■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 NQS Ⅱ (Network Queuing System Ⅱ)を用意しており、ジョブの操作は NQS Ⅱのコマンドを用い行います。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。演算 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ご利用ください。負担金の単価はバッチ処理に比べ割高に設定しています。SX-9 システムに直接ログインして実行します。

4-8 バッチ処理で実行する

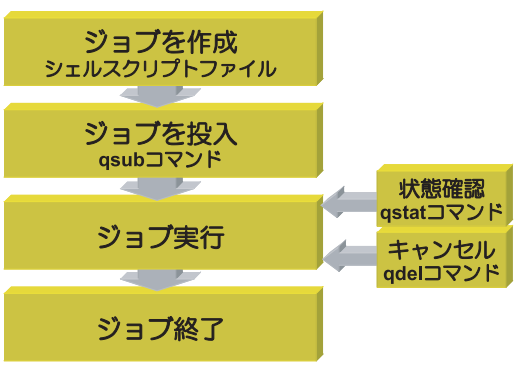
バッチ処理は並列コンピュータ上で作業を行います。

■4-8-1 バッチ処理の概要

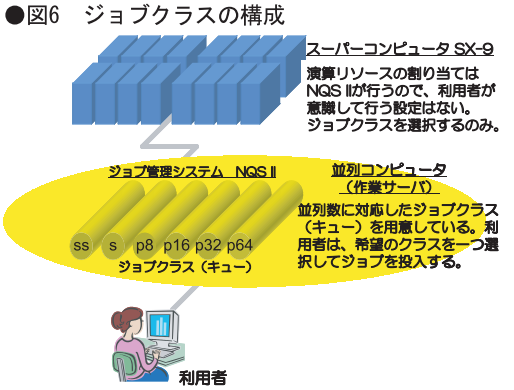
NQS Ⅱにプログラムの実行を依頼するため、実行の手続きを書いたジョブ（バッチリクエスト）を作成します。このジョブを NQS Ⅱに投入することで、プログラムの実行が可能になります。この際、NQS Ⅱではジョブクラスと呼ばれるキューが並列数に応じて設定してあり、利用者は目的のジョブクラスを 1 つ選択し投入します。順番が回ってくると NQS Ⅱは自動的にジョブを実行します。

ジョブ投入後は、ジョブの状態や込み具合の確認、また投入済みのジョブをキャンセルすることも可能です。プログラムの実行が終了するとジョブは NQS Ⅱ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 5　ジョブ（バッチリクエスト）の流れ



●図6　ジョブクラスの構成



■4-8-2 ジョブを作成

ジョブ投入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 スクリプトで記述します)。適当なファイル名を付け作成します。

リスト 2 はジョブファイルの一例で、ホームディレクトリ直下 work ディレクトリの a.out を実行する手続きを記述しています。

●リスト 2　ジョブファイル例

```
# test job.2          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
```

1 行目：# が先頭の行は、コメント行です。動作には影響しません。
2 行目：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。
3 行目：a.out はコンパイルして作成した実行形式ファイル名です。あらかじめコンパイルし作成しておきます。自動並列や OpenMP による並列処理も、同じ形式で指定します。

基本的には、ホームディレクトリから作業ディレクトリへ移動、そして実行する、の 2 行の設定です。これは最低限の設定例ですが、他に環境変数の指定、ファイルの操作コマンド等があれば適切な箇所に手続きを記述します。

以下の参考 1 と参考 2 は便利な記述方法です。

参考 1 ：作業ディレクトリの指定

NQS Ⅱ用の環境変数のひとつに 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。つまり、work ディレクトリでこのジョブを投入する（qsub を実行する）と、$PBS_O_WORKDIR にはカレントディレクトリの work が設定され cd work と同じことになります。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 3　ジョブファイル例
環境変数 PBS_O_WORKDIR

```
# test job.3
cd $PBS_O_WORKDIR     ディレクトリを環境変数で指定
a.out
```

参考 2 ：データファイルの指定

Fortran プログラムのファイル指定方法です。通常、ソース中 OPEN 文でファイル装置番号にファイル名を割り当てますが、環境変数 F_FF*nn* でファイル名を割り当てることもできます。(*nn* はファイル装置番号)

●リスト 4　ジョブファイル例
環境変数 F_FF*nn*

```
# test job.4
setenv F_FF11 datafile    装置番号 11 に、datafile を割り当てる。
cd $PBS_O_WORKDIR
a.out < infile > outfile
```

■4-8-3 ジョブを投入

プログラムの実行は、作成したジョブを NQS Ⅱに投入することで行います。

```
qsub オプション ジョブファイル
```

利用したい機能があれば、**表 5** に代表されるオプションを指定します。ここで、必ず指定するオプション-q があります。-q の後にジョブクラス名を指定することで、逐次処理または並列処理かを利用者が選択できるようになっております。本センターの SX-9 システムは、**表 6** のジョブクラスを用意しております。

●表5　主なオプション

-q　（必須）	投入するジョブクラス名を指定します。
-N	ジョブ名（リクエスト名）を指定します。指定がなければ、ジョブ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、ジョブ投入時のディレクトリに「ジョブ名.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、ジョブ投入時のディレクトリに「ジョブ名.e リクエスト ID」のファイル名で出力されます。
-jo	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	ジョブの実行が開始したときにメールが送られます。
-m e	ジョブ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	ジョブ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。

詳細は、man コマンド man qsub でご覧ください。

●表 6　ジョブクラス

ジョブクラス	利用可能 CPU 台数	経過 CPU 時間	メモリサイズ制限 (GBytes)
ss	4	1 時間	256
s	4	無制限	32
p8	8	無制限	512
p16	16	無制限	1024

単一 CPU で実行する逐次処理プログラムは s か ss クラスに投入します。ss クラスは時間制限がありますので、経過CPU時間が 1 時間以内の短いプログラムを投入します。並列化プログラムは並列数に応じて p8, p16 のいずれかに投入します。並列用のキュー(p8, p16)を利用するには、並列用のオブジェクト[5]を作成しておく必要があります。

s と ss クラスは逐次処理用のクラスですが、4 並列までの実行も可能です。これは並列化プログラムのデバッグを想定しており、特に ss クラスはメモリサイズを 256G バイトと大きく設定しています。

ジョブが正常に投入されると、システムから**リスト 5** のメッセージが返ります。1234.job がリクエスト ID で、ジョブの状況確認やキャンセル等、ジョブの操作の際に必要になります。

●リスト 5　qsub コマンド実行時のメッセージ

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

[5] 自動並列化なら-Pauto オプション、OpenMP なら-Popenmp オプションを付けてコンパイルしていること。

参考：オプションの埋め込み

毎回同じオプションを記述するのが面倒なとき、ジョブファイルに記述しておく方法もあります。

設定方法は、最初の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 6** の例は、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)、さらにジョブ名を reqname とする(-N)を、それぞれ指定しています。

また、埋め込みオプションとコマンド列に同じオプションを指定した場合は、コマンド列の方を有効とします。

●リスト 6　オプションの埋め込み

```
# test job.6
#PBS -q p8                    埋め込みオプション
#PBS -jo -N reqname             （複数）
cd $PBS_O_WORKDIR
a.out
```

■4-8-4 ジョブの状態確認

その１．自分のジョブ状態を確認する

```
qstat
```

qstat コマンドは、投入されたジョブの状態を表示します（**リスト 7**）。状態は STT（ステータス）項目に表示され、たいていは実行中(RUN)か実行待ち(QUE)のどちらかの状態になります。実行中の場合は、さらに CPU 時間、使用メモリサイズ等の情報が表示されます。また、リソースに空きがなければ実行待ち状態になり、順番が回ってくると自動的に実行状態に入ります。待ち状態中は、リクエスト ID の前に待ち順を表示します。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 8**）。

その２．システム全体のジョブ状態を確認する

```
qstat -Q
```

-Q オプションで、システム全体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、混雑具合いがわかります（**リスト 9**）。

●リスト 7　qstat コマンド例

```
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  p16       0  QUE  -    0.0B  0.0     0    Y Y Y    1
5:366.job        jobB     x2***9  p16       0  QUE  -    0.0B  0.0     0    Y Y Y    1
```

主な項目

RequestID	リクエスト ID 待ち状態(QUE)のリクエストは、先頭に待ち順の番号がつきます。
ReqName	ジョブ（リクエスト）名
UserName	ジョブの所有者
Queue	ジョブクラス（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)

●リスト 8　投入したジョブがない場合（ヘッダのみ）

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

●リスト 9　-Q オプションの例(一部抜粋)

```
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
p64               1    2 ENA ACT   32    4   0   0   2   0   2   0   0   0   0   0   0   0   0
p8                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
--------------- --- ---- ------- ---- ---------------------------------------------------------
```

主な項目

QueueName	ジョブクラス（キュー）名
TOT	ジョブ（リクエスト）の総数
QUE	待ちの件数
RUN	実行中の件数

■4-8-5 ジョブのキャンセル

投入済みのジョブを取り消します。

qdel リクエスト ID

●リスト 10　qdel コマンドの例

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

4-9　会話型処理で実行する

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。並列処理も 4 並列まで実行可能です。

手順は、一般的な UNIX を利用するように SX-9 にログインしコマンドラインで実行形式ファイル名を入力します（**リスト 11**）。CPU 時間と使用メモリサイズに制限があり、課金単価はバッチ処理より割高に設定しています（**表 7**）。

ホスト名

```
super.isc.tohoku.ac.jp
```

●リスト 11　会話型処理の例

```
yourhost$ ssh super.isc.tohoku.ac.jp -l 利用者番号
（パスワードを入力しログインする）

super$ a.out     実行
```

●表 7　会話型処理の制限値

演算 CPU 時間	1 時間
メモリサイズ	8 ギガ Bytes
利用可能 CPU 数	4

並列処理の場合、演算 CPU 時間は総 CPU 時間を対象にします。

5章　プログラミング
～ MPI 並列処理 ～

本章では、MPI による並列化プログラミング手順について紹介します。基本的な手順は前章と同じですので、コンパイルコマンド、利用できるジョブクラス等の異なる点について紹介します。

5-1 コンパイルする

MPI プログラムは、MPI 用コマンド sxmpif90 や sxmpicc コマンドでコンパイルします。

■MPI 並列 Fortran プログラムのコンパイル

sxmpif90 コマンドでコンパイルします。利用したい機能があれば適当なオプションと、Fortran ソースファイル名を指定します。

MPI 並列 Fortran プログラムをコンパイル

```
sxmpif90 オプション　ソースファイル名
```

オプションは、sxf90 コマンドと共通です。man sxf90 でご覧ください。

■MPI 並列 C プログラムのコンパイル

sxmpicc コマンドでコンパイルします。利用したい機能があれば適当なオプションと、C ソースファイル名を指定します。

MPI 並列 C プログラムをコンパイル

```
sxmpicc オプション　ソースファイル名
```

オプションは、sxcc コマンドと共通です。man sxcc でご覧ください。

5-2 バッチ処理で実行する

MPI プログラムの実行は、mpirun コマンドを用いますのでジョブファイルにもそのように記述します。

■ジョブを作成

MPI プログラムの実行コマンド

```
mpirun オプション　実行形式ファイル名
```

●表8　主なオプション

-np	MPI 並列数
-nn	使用ノード数

●リスト 12　ジョブファイルの例
64MPI 並列

```
# test job.12
cd $PBS_O_WORKDIR
mpirun -np 64 -nn 4 a.out    64 並列で実行
```

64MPI 並列より少ない並列数の場合、表 9 のオプションになります。-nn オプションを使用ノード数に合わせ設定します。

●表9　MPI 並列数とオプション

8 並列	mpirun -np 8 -nn 1 a.out
16 並列	mpirun -np 16 -nn 1 a.out
32 並列	mpirun -np 32 -nn 2 a.out
64 並列	mpirun -np 64 -nn 4 a.out

8 並列と 16 並列の場合、つまり使用ノード数が 1 のとき -nn 1 は省略できます。

■ジョブクラス

MPI プログラムは最大 64 並列まで実行可能です。64 並列は 4 ノード、32 並列は 2 ノードを占有して実行します。ノード内並列化プログラムと同様に 16 並列、8 並列も実行できます。ご希望の並列数のジョブクラスに投入します。

●表10　MPI プログラムのジョブクラス

ジョブクラス	利用可能 CPU 台数	経過 CPU 時間	メモリサイズ制限 (GBytes)
ss	4	1 時間	256
s	4	無制限	32
p8	8	無制限	512
p16	16	無制限	1024
p32	32	無制限	1024×2
p64	64	無制限	1024×4

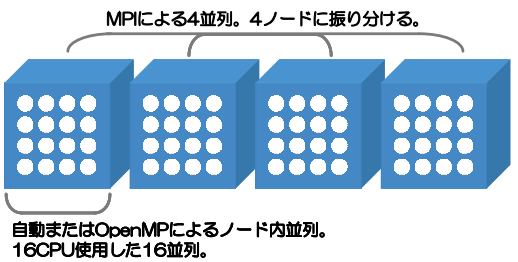
p64 クラス、p32 クラスのメモリサイズは、1 ノード(16CPU)当たり 1024G バイトに収まるようにしてください。その他、プログラムの実行手順は **4-8 バッチ処理で実行する** と同じです。

5-3 ハイブリッド並列

MPI と自動並列、または MPI と OpenMP を組み合わせたハイブリッド並列と呼ばれる並列化手法もあります (**図 7**)。通常、ノード間は MPI 並列化しノード内は自動並列化や OpenMP による並列化する形をとります。

例えば、64CPU を使ったハイブリッド並列を考えると、使用するノード数にあわせ 4 並列で動作する MPI プログラムを作成します。さらに、自動並列化オプション-Pauto をつけコンパイルすると、MPI と自動並列のハイブリッド並列用オブジェクトが作成されます。ノード間は MPI の 4 並列、ノード内は自動並列化の 16 並列、計 64 並列で実行できます。実行時のオプションは、-np で MPI 並列数の 4 を、-nn で使用ノード数 4 を指定します (**リスト 13**)。

●図 7　ハイブリッド並列　64 並列の例



MPI 4並列 × ノード内 16並列 ＝ 64並列

ハイブリッド並列化コンパイル(MPI+自動並列)

sxmpif90 -Pauto オプション ソースファイル名

ノード内が OpenMP 並列なら、-Popenmp オプションを使います。

ハイブリッド並列の実行コマンド

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 13　ジョブファイルの例
ハイブリッド並列

```
# test job.13
cd $PBS_O_WORKDIR
mpirun -np 4 -nn 4 a.out    4 ノード利用の
                            64 並列で実行
```

5-4 環境変数の設定 (p64, p32 クラス)

p64 または p32 クラスの実行で環境変数の設定を行う場合、環境変数 MPIEXPORT により各ノードに設定を反映させる必要があります。MPIEXPORT の設定がないと各 MPI プロセスに環境変数が反映されません。設定は、利用する環境変数名を MPIEXPORT の後に記述します(**リスト 14**)。

●リスト 14　環境変数 MPIEXPORT の指定例

```
# test job.14
#PBS -q p64
#PBS -jo -N reqname
setenv F_FF11 datafile
setenv F_FF12 datafile-B
setenv F_UFMTENDIAN  30,40
setenv MPIEXPORT       (次行へ続いています)
  "F_FF11 F_FF12 F_UFMTENDIAN"

cd $PBS_O_WORKDIR
mpirun -np 64 -nn 4 a.out
```

" "（ダブルクォーテーション）内に、空白で区切って記述します。

5-5 標準入出力ファイルを指定する

標準入出力ファイルは通常リダイレクション記号（<,>）で指定しますが、mpirun による MPI プログラムの実行では利用できません。

× 利用できません

```
mpirun -np 64 a.out < infile > outfile
```

この場合、標準入力ファイル指定は環境変数 F_FF05 で行います（**リスト 15**）。

標準出力および標準エラー出力は、MPI 並列数分複数出力されます。同一ファイルに入り混じり出力されないように、MPI プロセスごとにファイルを分け出力させます（**リスト 16**）。

●リスト 15　標準入力ファイルの指定例

```
# test job.15
#PBS -q p64
#PBS -jo -N reqname
setenv F_FF05 infile  標準入力は装置番号05
setenv MPIEXPORT "F_FF05"
cd $PBS_O_WORKDIR
mpirun -np 64 -nn 4 a.out
```

●リスト 16　標準出力、標準エラーファイルの指定例

```
# test job.16
#PBS -q p64
#PBS -jo -N reqname
setenv MPISEPSELECT 4
setenv MPIEXPORT "MPISEPSELECT"
cd $PBS_O_WORKDIR
mpirun -np 64 -nn 4     (次行へ続いています)
   /usr/lib/mpi/mpisep.sh a.out
```

シェルスクリプト /usr/lib/mpi/mpisep.sh を実行形式ファイル a.out の前に記述し、環境変数 MPISEPSELECT に 1～4 の値を指定することで、**表 11** の動作を選択します。

●表11　MPISEPSELECT 値と動作

MPISEPSELECT	動作
1	各 MPI プロセスの標準出力のみを stdout.uuu:rrr へ格納
2	各 MPI プロセスの標準エラーのみを stderr.uuu:rrr へ格納（規定値）
3	各 MPI プロセスの標準出力と標準エラーを各々stdout.uuu:rrr および stderr.uuu:rrr へ格納
4	各 MPI プロセスの標準出力と標準エラーを同一ファイル std.uuu:rrr へ格納

実行時、std.*や stderr.*の同名ファイルが存在する場合は、上書きでなく追加出力します。

5-6 MPI 用実行性能情報

MPIPROGINF 環境変数の設定により、プログラム性能情報（PROGINFO）の表示形式を変更することができます。DETAIL は Min/Max/Ave にまとめたもの、ALL_DETAIL は全てのランクについて、それぞれ性能情報を表示します。

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。全ランクの情報を出力します。

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　ジョブファイルの例
MPIPROGINF 環境変数

```
# test job.17
#PBS -q p64
#PBS -jo -N reqname
setenv MPIPROGINF DETAIL      実行性能情報
                               表示の指定
cd $PBS_O_WORKDIR
mpirun -np 64 -nn 4 a.out
```

6章　その他

使用メモリサイズ

実行に必要なメモリサイズを、実行前に確認することができます。ただし allocate 等で動的に確保するメモリサイズは含まれません。

このコマンドは SX-9 上で実行します。

逐次プログラムの場合

```
size 実行形式ファイル名
```

並列化プログラムの場合
（自動並列または OpenMP）

```
size -fl 並列数 実行形式ファイル名
```

●リスト 18　size コマンド表示例

```
super$ size a.out.s
1046912 + 140272 + 418928 = 1606112
```

逐次プログラム a.ous.s を実行すると、1,606,112 バイト使用します

```
super$ size -fl 16 a.out.p
1046912(.text) + 140272(.data) + 418928(.bss) + 181855(.comment) + 4496(.whoami)
+ 1048576(logical task region) * 16 = 18569679
```

並列化プログラム a.out.p を 16 並列実行すると、18,569,679 バイト使用します

バイナリファイルの扱い　[Fortran]

データファイルを入力するなど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。SX-9 システムはBig-Endian仕様ですので、Little-Endian仕様[6]のバイナリファイルを扱うには、Endianを合わせる必要があります。変換は環境変数F_FUFMTENDIANを用い、Big-Endianに変換したいLittle-Endianファイルの装置番号を指定します。

環境変数 F_FUFMTENDIAN

```
setenv F_UFMTENDIAN u[,u]…
```

u は Little-Endian ファイルの装置番号です。複数ある場合は、,（カンマ）で区切って羅列します。

●リスト 19　ジョブファイル例

```
# test job.19
setenv F_UFMTENDIAN 30,40
cd $PBS_O_WORKDIR
a.out
```

装置番号 30,40 を Big-Endian に変換する

バウンズチェック　[Fortran]

配列の添字について、宣言している範囲内を使っているかを検査します。

プログラムの実行中に、適切な値を処理しているはずなのにエラー番号 250(オーバーフロー)や 252(ゼロ割り)、253(演算例外)等が発生する、または実行エラーは発生しないが実行する毎に動作や演算結果が異なるという場合、配列の添字検査をしてみてください。配列で参照されている添字が許される範囲にあるかどうかを調べ、範囲外であれば配列の大きさ、添字の値を表示します。

チェック方法は、-eC オプションを付けコンパイルしてから実行します。範囲外添字が見つかると、標準エラー出力ファイルにエラーメッセージを出力します（リスト 20）。

バウンズチェックのオプション

```
sxf90 -eC ソースファイル名
```

-eC オプションを指定すると、最適化やベクトル化および並列化はされませんので通常の実行に比べ時間がかかります。添字検査を行う時のみ、このオプションを用いてください。

[6] プロセッサによって仕様が異なります。ちなみに、本センターの並列コンピュータシステムは、Intel 製 Itanium2 なので Little-Endian 仕様です。

●リスト 20　エラーメッセージ例

```
*240  Subscript  error  array=b  size=10  subscript=11  eln=756  PROG=main
ELN=756(400021981)
```

配列 b はサイズ 10 の宣言であるが、添字 11 を使っているためエラーが発生している。
エラー発生箇所は、main ルーチン 756 行目。

パスワードの変更

ログイン名とパスワードは、スーパーコンピュータシステム、並列コンピュータシステムで共通です。初めてご利用の場合、初期パスワードが設定されていますので、ログイン後 yppasswd コマンドで速やかに変更してください（**リスト 21**）。並列コンピュータでコマンドを実行します。

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勧めします。

●リスト 21　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
```

ログインシェルの変更

ログインシェルは､規定値で csh を設定しています。tcsh 等に変更したい場合は ypchsh コマンドを実行後、再ログインしてください（**リスト 22**）。

並列コンピュータでコマンドを実行します。

●リスト 22　ログインシェルの変更

```
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.

（一度ログアウトして、再ログインする）
```

マニュアル

冊子マニュアルは、本センター本館 1 階 利用者相談室に備えております。

■冊子

[1]FORTRAN90/SX 言語説明書
[2]FORTRAN90/SX プログラミングの手引
[3]FORTRAN90/SX 並列処理機能利用の手引
[4]MPI/SX 利用の手引
[5]C++/SX プログラミングの手引
[6]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）
[7]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）
[8]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）
[9]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）
[10]科学技術計算ライブラリ ASL/SX 利用の手引
　（高速機能編）
[11]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）

■オンライン

冊子番号[1]～[4]についてはオンラインマニュアルも用意しております。並列コンピュータ上で mansxf90 コマンドを実行します。HTML 形式のマニュアルがご覧になれます。

```
mansxf90
```

利用負担金

利用負担金は、主に演算負担経費とファイル負担経費からなります。演算負担経費は、演算した CPU 時間に応じて課金され、ファイル負担経費は週一回集計するファイル使用量を基に算出されます。

請求書は 4 半期（3 ヶ月）ごと[7]に、利用者を取りまとめている支払い責任者に発行します。

また、平成 20 年度も利用者全員に適用される利用負担金割引制度があります。演算した分だけ割引率が上がり、長時間利用するほどお得な制度になっ

[7] 年度末は、2 月中旬にも請求しています。

ています。詳しくは、以下の Web サイトをご覧ください。

http://www.cc.tohoku.ac.jp/　“利用負担金”

●表12　負担額単価表

		負担額
演算負担経費	スーパーコンピュータ	バッチ処理　0.4 円／秒
		会話型処理　2　円／秒
	並列コンピュータ	バッチ処理　0.1 円／秒
		会話型処理　0.2 円／秒
ファイル負担経費		1M バイト　0.1 円／日

並列演算の場合、最長の CPU 時間を演算時間とします。

現在の利用額を表示するには、以下 2 つのコマンドがあります。

■利用額の表示【利用者】

```
kakin
```

前日までの利用額を表示します（**リスト 23**）。kakin コマンドを実行した個人の利用額です。並列コンピュータで実行します。

■利用額の表示【支払責任者】

```
skakin
```

前日までの利用額と負担額を、支払い責任者単位で表示します（**リスト 24**）。利用額は実際利用した額、負担額は利用額から割引制度で減算した額になります。負担額の方を四半期ごとにご請求します。

このコマンドも並列コンピュータで実行します。

●リスト 23　kakin コマンド表示例

```
利用者番号=x2xxxx
                    利 用 額
月    スーパー    並 列    ファイル    そ の 他    調 整 額    合 計
 4        2262      592       58286           0           0    61140
 5           6        8       19421           0           0    19435
 6           0        0           0           0           0        0
 7           0        0           0           0           0        0
 8           0        0           0           0           0        0
 9           0        0           0           0           0        0
10           0        0           0           0           0        0
11           0        0           0           0           0        0
12           0        0           0           0           0        0
 1           0        0           0           0           0        0
 2           0        0           0           0           0        0
 3           0        0           0           0           0        0
合計      2268      600       77707           0           0    80575
```

●リスト 24　skakin コマンド表示例

```
6月 12日 現在の利用額および負担額は次のとおりです。
  支払責任者 ： 東北　太郎                          (u20000)

                合計          演算        ファイル        出力      その他
利用額     2,159,948     1,443,157         711,391       5,400           0
負担額     1,038,370       321,579         711,391       5,400           0
-------------------------------------------------------------------------
  月別利用額
       4月          1,597,898
       5月            562,050
-------------------------------------------------------------------------
  利用者別利用額
       x10000         123,456
       x10001               0
       x10002       1,656,431
```

お知らせ、お問い合わせ

■システム運用に関するお知らせ
大規模科学計算システム
http://www.cc.tohoku.ac.jp/

利用方法やシステムの運用について最新情報をお知らせします。

■システム利用に関するお問い合わせ
利用相談室　　sodan05@isc.tohoku.ac.jp
システム管理係　　022-795-6252

■システム利用申請
詳細は、以下のウェブページをご参考ください。
http://www.cc.tohoku.ac.jp/guide/riyou.html

窓口（庶務係）　　uketuke@isc.tohoku.ac.jp
022-795-3406

7章　おわりに

本稿は、SX-9 システムのプログラミング利用ガイドとして基本的な手順を紹介しました。

研究室のサーバでは実現できなかったプログラムやアイデアを、ぜひ最新鋭のスーパーコンピュータシステムでお試しください。研究の強力なツールとして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、お気軽にセンターまでお問い合わせください。