［大規模科学計算システム］

並列コンピュータ LX 406Re-2 の利用法

情報部情報基盤課　共同利用支援係　共同研究支援係
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

本センターは並列コンピュータ LX 406Re-2 の運用を 2014 年 4 月から開始しています。本稿では、LX 406Re-2 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、実行等の使い方と利用負担金についてご紹介します。

2 章　システム構成

システム構成は、既設のシステムを含め図1のようになっています。

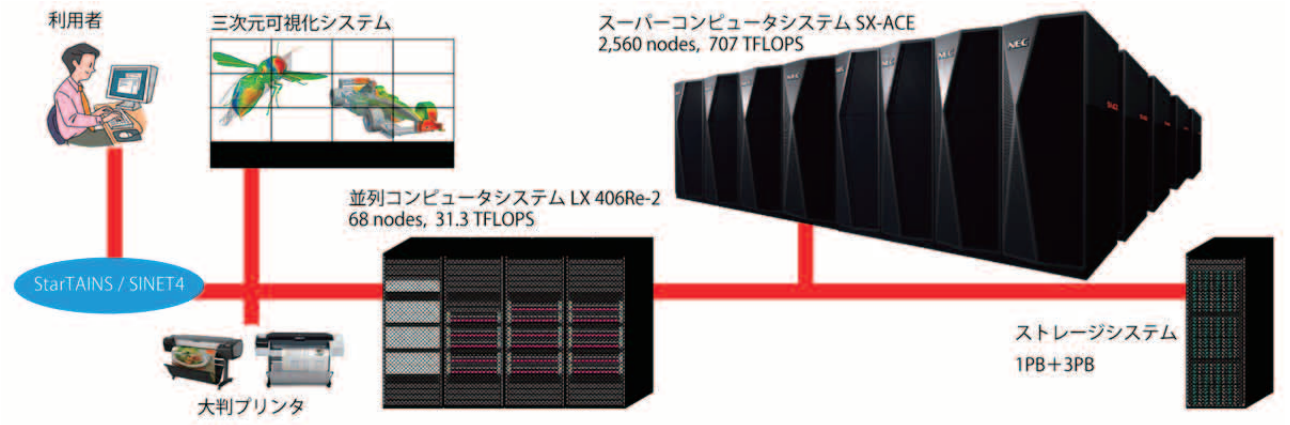

図 1. 大規模科学計算システム構成図

並列コンピュータ LX 406Re-2

並列コンピュータ LX 406Re-2 は 1 ノードに、インテル Xeon プロセッサ E5-2695v2（12 コア）を 2 基と 128GB の主記憶装置を搭載し、合計 68 ノードで構成されます。自動並列化・OpenMP・MPI を利用したノード内の並列処理は 24 並列まで可能で、ノードあたりの最大演算性能は 460.8GFLOPS（倍精度）となります。複数のノードを使用した並列処理は、MPI の利用により最大 576 並列まで実行可能です。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-ACE のフロントエンドサーバとしての役割も担っています。

3 章　プログラミング　～逐次処理、共有メモリ並列処理 ～

本章では、単一のコアで実行する逐次処理と、自動並列化および OpenMP による共有メモリ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、4 章で紹介しますが、コンパイルコマンド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。

■ 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[1]をご利用ください。

並列コンピュータの OS は Linux です。公開鍵暗号方式による認証のみ利用できます[2]。アカウント希望の場合は、共同利用支援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。

並列コンピュータへの初回ログイン時には公開鍵と秘密鍵のペアを作成する必要があります。鍵ペアの作成方法については本誌 105 ページの「SSH アクセス認証鍵生成サーバの利用方法」をご参照ください。

なお、他人名義の利用者番号でのシステム利用は禁止します。パスワード、秘密鍵、パスフレーズの使い回しは、不正アクセスのリスク（不正ログイン、クライアントのなりすまし、暗号化された通信の暴露、他サーバへの攻撃等）が非常に高く、大変危険です。利用者登録を行うことによる年間維持費等は発生しませんので、利用される方はそれぞれで利用申請をお願いいたします。

並列コンピュータホスト名

```
front.cc.tohoku.ac.jp
```

リスト 1. ssh コマンドによる接続例

```
localhost$ ssh -i ~/.ssh/id_rsa 利用者番号@front.cc.tohoku.ac.jp
Enter passphrase for key '/home/localname/.ssh/id_rsa':パスフレーズを入力
（初回接続時のメッセージ） : yes を入力
Front1$ （コマンド待ち状態）
```

暗号鍵は複数の端末や他人と共有してはいけません。また、メールに暗号鍵を添付したり USB メモリにコピーしたりすると不正アクセスのリスクとなります。並列コンピュータにログインする端末を追加する場合は、その端末で新規に鍵ペアを作成し、authorized_keys に追加登録してください。

■ ログイン端末の追加方法

ログインする端末を追加する場合は、追加する端末で鍵ペアの作成を行います。必ずパスフレーズの設定を行って鍵を作成して下さい。作成した公開鍵を並列コンピュータにログイン出来る端末に転送します。公開鍵ですので転送方法はメール本文に記載しても構いません。

作成した公開鍵の内容を利用者のホームディレクトリのファイル（~/.ssh/authorized_keys）に追記します。接続元が Linux、OS X の場合の鍵の作成方法をリスト 2 に示します。

[1] Windows であれば、TeraTerm 等のフリーソフトが利用できます。
[2] パスワード認証方式は 2015 年 4 月 13 日で廃止しました。

リスト 2. 公開鍵と秘密鍵の作成方法

```
【利用する端末で鍵ペアを作成】
localhost $ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/localname/.ssh/id_rsa):（ファイル名を指定）

Enter passphrase (empty for no passphrase):（必ずパスフレーズを設定）
Enter same passphrase again:（同じパスフレーズを入力）

指定した場所（/home/localname/.ssh）に鍵ペア（暗号鍵：id_rsa　公開鍵：id_rsa.pub）が生成される

【作成した公開鍵を既に並列コンピュータにログイン出来る端末に転送】

【作成した公開鍵を並列コンピュータに転送】
localhost $ scp /home/localname/.ssh/id_rsa.pub 利用者番号@front.cc.tohoku.ac.jp:~
（パスフレーズを入力）

【公開鍵を追記登録】
front1 $ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
front1 $ exit
```

~/.ssh/authorized_keys を削除すると、全ての暗号鍵からのログインが出来なくなります。センターではセキュリティインシデントに対する緊急対応として、全ユーザの~/.ssh/authorized_keys を削除する場合があります。

■ パスワードの変更

パスワードの変更は passwd コマンドで行います。パスワードの変更方法をリスト 3 に示します。

コマンドを実行すると、フロントエンドサーバ（並列コンピュータ）、可視化サーバ、プリンタサーバ、およびファイル転送サーバのログインパスワードが変更されます。入力したパスワードは表示されません。

リスト 3. パスワードの変更方法

```
front1 $ passwd
ユーザー 利用者番号 のパスワードを変更。
Enter login(LDAP) password:  （現在のパスワードを入力）
新しいパスワード:  （新しいパスワードを入力）
新しいパスワードを再入力してください:（新しいパスワードを入力）
LDAP password information changed for 利用者番号
passwd: 全ての認証トークンが正しく更新できました。
```

■ ログインシェルの確認と変更

ログインシェルの確認と変更は fchsh コマンドで行います。ログインシェルの確認方法と変更方法をリスト 4 に示します。

ログインシェルの変更が SX-ACE と LX 406Re-2 に反映されるまで 15 分程度かかります。

リスト 4. ログインシェルの確認方法と変更方法

```
front1 $ fchsh （ログインシェルの確認）
Enter Password: （パスワードを入力）
loginShell: /bin/tcsh （現在のログインシェルが表示される）

front1 $ fchsh /bin/bash （ログインシェルを/bin/bash に変更）
Enter Password: （パスワードを入力）
Changed loginShell to /bin/bash （ログインシェルが変更された）
```

■ 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。利用者番号作成時の容量制限は 1TB です。ファイル容量の追加申請によりディスク領域を増やすことも可能です。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。

```
/uhome/利用者番号
```

■ 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。

表1．プログラミング言語およびライブラリ

Fortran	Intel Fortran Composer XE
C/C++	Intel C++ Composer XE
MPI	Intel MPI ライブラリ
数値演算ライブラリ	NEC NumericFactory, Intel MKL 他

■ ファイルエディット

ソースファイルは、並列コンピュータにログインし、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、front.cc.tohoku.ac.jp の利用者ディレクトリ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、以下の Web ページをご参照ください。

http://www.ss.cc.tohoku.ac.jp/application/setting.html

■ コンパイル

Fortran および C/C++コンパイラの基本的な使用方法です。詳しいオプション等については man コマンド、および**マニュアル**をご覧ください。

Fortran プログラムのコンパイル

ifort 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、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。

● コンパイル【逐次処理】

```
front1 $ ifort  オプション   ソースファイル名
```

● コンパイル【自動並列化】

```
front1 $ ifort  -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。

主なオプション

-parallel -par-report	自動並列化機能を利用する。 自動並列化されたループの行番号を表示する。
-openmp -openmp-report	OpenMP を利用する。 OpenMP 指示行により並列化されたループ、領域、セクションの行番号を表示する。
-O0	最適化を無効にする。
-O1	最適化を行うが、コードサイズが増える最適化は行わない。
-O2 または -O	一般的な最適化を行う。（規定値）
-O3	高度の最適化（プリフェッチ、スカラリプレスメント、ループ変換等）を行う。
-ip	インライン展開を行う。
-c	コンパイルのみ行う。（リンクはしない）
-o	実行可能形式のオブジェクトファイルの名前を指定する。省略時は a.out になる。
-w90	非標準 Fortran 機能に関する警告メッセージを抑止する。
-r8	精度の自動拡張を行う。（倍精度化）
-help	オプションの種類と説明を表示する。

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。

C/C++プログラムのコンパイル

C プログラムを **icc** コマンドで C++プログラムを、**icpc** 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、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。実行する時点で希望する並列数を環境変数で指定します。ノード内並列数は自動並列の場合は環境変数 F_RSVTASK で指定し、OpenMP 並列の場合は環境変数 OMP_NUM_THREADS で指定します。詳細は 5 章で説明します。

● コンパイル【逐次処理】

```
front1 $ icc  オプション  ソースファイル名
front1 $ icpc  オプション  ソースファイル名
```

● コンパイル【自動並列化】

```
front1 $ icc -Pauto  オプション  ソースファイル名
front1 $ icpc -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。

主なオプション

-parallel -par-report	自動並列化機能を利用する。 自動並列化されたループの行番号を表示する。
-openmp -openmp-report	OpenMP を利用する。 OpenMP 指示行により並列化されたループ、領域、セクションの行番号を表示する。
-O0	最適化を無効にする。
-O1	最適化を行うが、コードサイズが増える最適化は行わない。
-O2 または -O	一般的な最適化を行う。（規定値）
-O3	高度の最適化（プリフェッチ、スカラリプレスメント、ループ変換等）を行う。
-ip	インライン展開を行う。
-c	コンパイルのみ行う。（リンクはしない）
-o	実行可能形式のオブジェクトファイルの名前を指定する。省略時は a.out になる。
-help	オプションの種類と説明を表示する。

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++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。

4 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の異なる点について紹介します。

■ コンパイルを行う

MPI プログラムは、MPI 用コマンド mpiifort、mpiicc、mpiicpc 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

MPI 並列の Fortran プログラムは mpiifort コマンドでコンパイルします。利用したい機能があればオプションと、ソースファイル名を指定します。

```
front1 $ mpiifort  オプション  ソースファイル名
```

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

MPI 並列 C/C++プログラムのコンパイル

MPI 並列の C プログラムは **mpiicc** コマンドで、MPI 並列の C++プログラムは **mpiicpc** 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、ソースファイル名を指定します。

```
front1 $ mpiicc  オプション  ソースファイル名
front1 $ mpiicpc オプション  ソースファイル名
```

・オプションは、**icc** コマンド、**icpc** コマンドと共通です。詳細は man icc コマンド、man icpc コマンドでご確認ください。

5 章 バッチリクエスト

■ 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。

■ バッチ処理

プログラムの実行は、NQSⅡのコマンドを用いて操作します。図2は作業の流れを示しています。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バッチリクエストを作成します。このバッチリクエストを NQSⅡに投入することで、プログラムの実行が可能になります。

バッチリクエストの投入後は、バッチリクエストの状態や、混み具合の確認、また投入済みのバッチリクエストをキャンセルすることも可能です。プログラムの実行が終了するとバッチリクエストは NQSⅡ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。

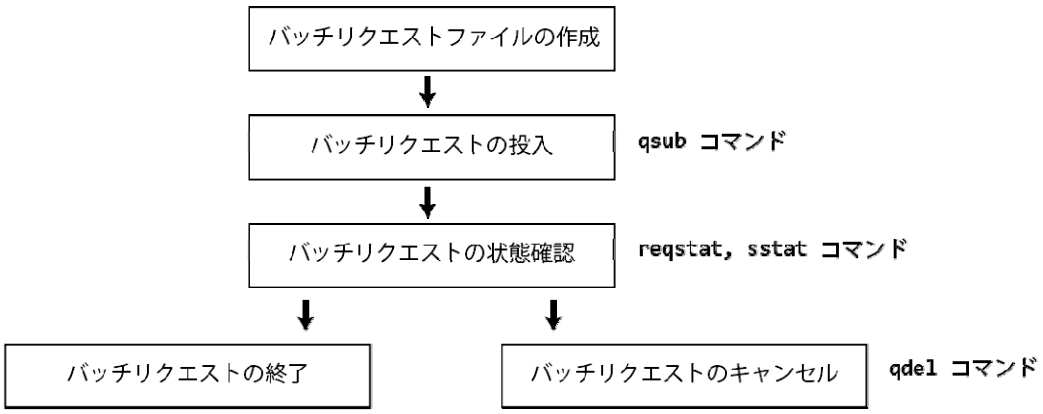

図 2. NQSII によるバッチリクエストの流れ

【バッチリクエストファイルの作成】

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。以降ではバッチリクエストファイル名を run.csh とします。

基本的に必要となるのは、実行するマシンとノード数の指定、ホームディレクトリから作業ディレクトリへ移動、プログラムの実行、です。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。

並列コンピュータ LX 406Re-2 で実行する場合の利用形態と必須オプションを表 2 に示します。通常の利用の場合、-q の後に lx を指定し、-b の後に利用ノード数または a を指定します。

実行時間の設定は、-l elapstim_req=hh:mm:ss で設定します。通常利用で 1～24 ノードを利用する場合、実行時間制限の規定値でリクエストは自動的に終了します。実行時間が規定値を超えるリクエストは、必ず実行時間を指定してください。実行時間は最大値まで設定が可能です。その他のオプションは表 3 をご参照ください。

表 2. 並列コンピュータ LX 406Re-2 の利用形態と-q および-b オプション

利用形態	利用ノード数	実行時間制限 （経過時間）	メモリサイズ制限	-q オプション	-b オプション
通常	1～24	規定値：1 カ月 最大値：1 カ月	128GB×ノード数	lx	利用ノード数
ｱﾌﾟﾘｹｰｼｮﾝ	1	なし	128GB	lx	a

表 3. qsub コマンドの主なオプション

-q（必須）	計算機名 lx を指定します。
-b（必須）	リクエストを実行するノード数、または a を指定します。
-A	課金先のプロジェクトコードを指定します。指定が無ければデフォルトのプロジェクトコードに課金されます。
-N	リクエスト名を指定します。指定がなければ、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。
-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 elaptim_req=hh:mm:ss	最大経過時間を指定します。設定時間は、時:分:秒を hh:mm:ss の形式で指定します。
-m b	リクエストの処理が開始したときにメールが送られます。
-m c	リクエストの処理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	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@front.cc.tohoku.ac.jp」宛に送られます。

・その他オプションの詳細は、man qsub コマンドでご覧ください。

● 逐次プログラム、自動並列/OpenMP 並列の場合

リスト 5 は逐次プログラムを実行する場合のバッチリクエストファイルの一例です。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 5. 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                         #実行形式ファイルを指定
```

・1 行目 :#以降はコメントです。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。

● 作業ディレクトリの指定

NQSⅡ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（リスト 6）。つまり、work ディレクトリでこのバッチリクエストを投入する(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 6. 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                  コメント行
cd $PBS_O_WORKDIR              #作業ディレクトリを環境変数で指定
./a.out                        #実行形式ファイルを指定
```

● 実行時のデータファイル指定

Fortran プログラムで入出力ファイルを割り当てる環境変数 FORT*n* です。n が 1～9 の場合には 0 をつけず 1 桁で指定します（リスト 7）。
正しい指定方法:　　setenv FORT2 datafile

リスト 7. 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                   コメント行
setenv FORT1 datafile          #装置番号 1 に、ファイル datafile を割り当てる
cd $PBS_O_WORKDIR              #作業ディレクトリを環境変数で指定
./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
```

● 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 の例は、2 行目で実行計算機に lx を指定(-q lx)、利用ノード数 6 を指定(-b 6)、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し(-jo)、4 行目でリクエスト名を test04 とする(-N test04)を、それぞれ指定しています。

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、コマンド列の方が優先されます。

リスト 8. run.csh 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 lx -b 6                #実行マシンとノード数を指定
#PBS -jo                       #標準エラー出力を標準出力と同じファイルへ出力
#PBS -jo -N test04             #リクエスト名を test04 にする
cd $PBS_O_WORKDIR              #作業ディレクトリを環境変数で指定
./a.out
```

・【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ファイルを NQSⅡに投入することで行います。

```
front1 $ qsub オプション　バッチリクエストファイル名
```

リクエストが正常に投入されると、システムからのメッセージが返ります（リスト 9）。1234.job1 がリクエスト ID で、リクエストの状況確認やキャンセル等、リクエストの操作の際に指定が必要になります。

リスト 9. qsub コマンドの実行例

```
front1$ qsub run.csh
Request 1234.job1 submitted to queue: lx6.
```

フラット MPI プログラムの実行（MPI のみの並列）

MPI プログラムは、mpirun コマンドを使用して実行します。バッチリクエストファイルに mpirun コマンドとオプションを記述します（リスト 10）。

-ppn オプションに **1 ノードあたりのプロセス数**を指定します。-np オプションに**合計プロセス数**を指定します。

-ppn オプションは-np オプションよりも前で指定する必要があります。

表 4. mpirun コマンドのオプション（必須）

オプション	引数
-ppn	1 ノードあたりのプロセス数
-np	合計プロセス数を指定

リスト 10. フラット MPI プログラム用バッチリクエストファイル例

（MPI 並列数 144 で実行する場合）

```
# test job-a
#PBS -q lx -b 6                    #実行マシンとノード数を指定
cd $PBS_O_WORKDIR                  #作業ディレクトリを環境変数で指定
mpirun -ppn 24 -np 144 ./a.out     #MPI プログラムの実行
```

ハイブリッド並列プログラムの実行（MPI と自動並列/OpenMP を組み合わせた並列）

ハイブリッド並列プログラム実行時の並列数は「プログラム並列数＝MPI 並列数×SMP 並列（自動並列/OpenMP）」になります。MPI プログラムの並列数は mpirun コマンドの -ppn オプションと -np オプションで制御し、自動並列/OpenMP 並列数は OMP_NUM_THREADS 環境変数で制御します（リスト 11）。

表 5. ノード内並列数を指定する環境変数

並列方法		環境変数
自動並列（-Pauto）	Fortran プログラムの場合	F_RSVTASK
	C/C++プログラムの場合	C_RSVTASK
OpenMP 並列（-Popenmp）		OMP_NUM_THREADS

リスト 11. ハイブリッド並列プログラム用バッチリクエストファイル例

（MPI 並列数 6、自動並列数/OpenMP 並列数 24 の 144 並列で実行する場合）

```
# test job-a
#PBS -q lx -b 6                    #埋め込みオプション
setenv OMP_NUM_THREADS 24          #自動並列/Open MP での並列数
cd $PBS_O_WORKDIR                  #作業ディレクトリを環境変数で指定
mpirun -ppn 1 -np 6 ./a.out        #MPI プログラムの実行
```

■ バッチリクエストの状態確認

● reqstat コマンド

reqstat コマンドは、投入されたリクエストの状態を表示します（リスト 12）。状態は STATE 項目に表示されます（表 7）。リクエストはジョブサーバ（job1 または job2）毎に出力されます。

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。システム内に自分のリクエストが存在しない場合は、「No request.」と表示されます（リスト 13）。

リスト 12. reqstat コマンド例

```
front1$ reqstat
statistics sampled at 2015/02/20 19:18:01 in job1.
REQUEST ID      USER     GROUP    QUEUE    T NODE ELAPS     STATE       TIMES              REQUEST NAME
--------------- -------- -------- -------- - ---- --------- ----------- ------------------ ---------------
2512.job1       利用者番号 users    lx6      S   6    1:00:00 running     15/02/20 12:00:00 test02
2513.job1       利用者番号 users    lx6      S   1    1:00:00 queued      15/02/20 18:55:00 test04
```

表 6. reqstat コマンドの主な表示項目

項目名	内容
RequestID	リクエスト ID
USER	利用者番号
GROUP	利用者の所属グループ
QUEUE	キュー名
T	リクエストタイプ（通常は S）
NODE	利用ノード数
ELAPS	経過時間制限
STATE	リクエストのステータス
TIMES	ステータスが変化した日時
REQUEST NAME	-N で指定したリクエスト名もしくはバッチリクエストファイル名

表 7. STATE 項目の主な表示とリクエスト状態

表示	リクエスト状態
wait	実行ノードの決定待ち
queued	実行ノードが決まり、実行順待ち
running	実行中

リスト 13. 投入したリクエストがない場合

```
front1 $ reqstat
No request.
```

● sstat コマンド

sstat コマンドは投入されたリクエストの実行開始予定時刻を表示します（リスト 14）。ただし、マップスケジュール機能が有効な場合のみです。エスカレーション機能により、予定実行開始時刻が他の待ちリクエストより早まることがあります。-l elapstim_req オプションで指定した時間が短いほど、待ちリクエストの隙間にリクエストの実行が割り当てられる可能性が大きくなりますので、必要十分な実行時間を指定することをお勧めいたします。

リスト 14. sstat コマンド例

```
front1 $ sstat

RequestID       ReqName  UserName Queue                  Pri STT PlannedStartTime
--------------- -------- -------- -------- ----------------- --- -------------------
2512.job1       test03   利用者番号 lx6       0.5002/  0.5002 RUN Already Running...
2513.job1       test04   利用者番号 lx6       0.5002/  0.5002 ASG 2015-02-20 19:22:20
```

表 8. sstat コマンドの表示項目

項目名	内容
RequestID	リクエスト ID
ReqName	-N で指定したリクエスト名
UserName	利用者番号
QUEUE	キュー名
Pri	優先度
STT	リクエストのステータス
PlannedStartTime	予定実行開始時刻

■ バッチリクエストのキャンセル

投入したリクエストの削除、または実行中のリクエストを停止する場合は、qdel コマンドにリクエスト ID を指定します(リスト 15)。リクエスト投入時、または reqstat コマンドで表示されるリクエスト ID をジョブサーバ名まで指定してください。

リスト 15. qdel コマンド例

```
front1 $ qdel 1234.job1
Request 1234.job1 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、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 16)。表 9 は会話型処理の制限値です。時間制限は CPU 時間の合計ですので、並列実行した場合はそれぞれの CPU 時間の合計値となり、1 時間経過する前にジョブが終了します。

リスト 16. 会話型処理の例(a.out を実行する)

```
yourhost$ ssh front.cc.tohoku.ac.jp -l 利用者番号      front にログインする
    :
front1$ a.out
（プログラム実行中）
front1$
（実行終了）
```

表 9. 会話型処理の制限値

利用ノード数（最大並列数）	時間制限 [時間]	最大メモリ [GB]
1(6)	1 時間（CPU 時間合計）	8

6 章　ライブラリ

以下のライブラリを使用することができます。

Fortran,C/C++ 用

数値計算ライブラリ集	NEC NumericFactory
数値演算ライブラリ	Intel MKL
画像処理ライブラリ	Intel IPP
マルチスレッドライブラリ	Intel TBB

■ 数値計算ライブラリ集　NEC NumericFactory

【機能概要】

NumericFactory は、NEC が独自に開発している数値計算ライブラリと、数値シミュレーションプログラムで頻繁に利用される OSS（Open Source Software）により、多彩な数値計算アルゴリズムを提供します。NumericFactory の使用により、プログラム開発の時間を短縮でき、高品質なプログラムを開発することが出来ます（表 10）。

NumericFactory は、全 12 種類のライブラリで構成されています。ライブラリにより、使用できる言語が異なります（表 11）。また、並列化された機能を含むものと含まないものがあります。OpenMP 並列の機能がないライブラリでも、下位で使用する Intel MKL が並列化されている場合、マルチスレッドで動作することがあります。OpenMP 並列、または、MPI 並列機能がないライブラリでも、OpenMP/MPI プログラムから利用することは可能です。

表 10. NumericFactory の機能概要

ライブラリ名	機能概要
ASL	行列積、疎行列用連立 1 次方程式（直接法／反復法）、固有値方程式、FFT、乱数、特殊関数、近似・補間、スプライン、微分方程式、数値微積分、方程式の根、数理計画法、ソート・順位付け
ASLSTAT	乱数、基礎統計量、推定・検定、分散分析・実験計画、多変量解析、フーリエ解析、回帰分析
ASLQUAD	四倍精度演算機能（基本演算、連立 1 次方程式、固有値方程式、特殊関数）
SFMT	メルセンヌツイスター擬似乱数生成（整数）
dSFMT	メルセンヌツイスター擬似乱数生成（実数）
SuperLU	疎行列用連立 1 次方程式（直接法）
MUMPS	疎行列用連立 1 次方程式（直接法）
Lis	疎行列用連立 1 次方程式、疎行列用固有値方程式（反復法）
ARPACK	大規模固有値問題
PARPACK	大規模固有値問題（MPI 版）
XBLAS	精度拡張／精度混合行列積
METIS	行列、グラフ並べ替え、グラフ分割

表 11. NumericFactory のライブラリと利用可能言語

ライブラリ名	Fortran から利用	C から利用	OpenMP 並列機能	MPI 並列機能
ASL	○	○	○	×
ASLSTAT	○	○	×	×
ASLQUAD	○	×（C++可）	○	×
SFMT	×	○	×	×
dSFMT	×	○	×	×
SuperLU	×	○	×	×
MUMPS	○	○	×	○
Lis	○	○	○	○
ARPACK	○	×	×	×
PARPACK	○	×	×	○
XBLAS	○	○	×	×
METIS	○	○	×	×

【ライブラリのリンク方法】

● 逐次版/OpenMP 版プログラムの場合

各ライブラリのリンクには、ifort、icc コマンドを使用します。利用するプログラム言語に応じて、リスト 17、18 のようにリンクしてください。リンクオプションは使用するライブラリと言語に応じて指定します（表 12、表 13）。

さらに今回から、MKL のリンクオプションが，以下のように簡潔に指定することもできるようになりました。

-lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -pthread

↓

-mkl=sequential

詳細は、man コマンドでご覧ください。

NumericFactory でサポートしているライブラリを使用する場合、ライブラリによってはユーザプログラム側でモジュールファイルやヘッダファイルをインクルードする必要があります（表 14）。

Fortran から ASL または ASLSTAT の 64 ビット整数に対応したライブラリを利用する場合、コンパイル時に必ずオプション"-i8"を付けてコンパイルしてください。このオプションは、integer 型を 64 ビット整数と翻訳する Intel コンパイラのオプションです。

リスト 17. Fortran の場合（逐次版/OpenMP 版）

```
[front1 ~]$ ifort source.f90 <リンクオプション>
```

リスト 18. C の場合（逐次版/OpenMP 版）

```
[front1 ~]$ icc source.c <リンクオプション>
```

表 12. NumericFactory のリンクオプション（逐次版/OpenMP 版 Fortran プログラムから利用する場合）

ライブラリ名	リンクオプション	
ASL	32bit整数/逐次版	-lasl -mkl=sequential
	64bit整数/逐次版	-lasl64 -mkl=sequential
	32bit整数/OpenMP版	-lasl -mkl=parallel
	64bit整数/OpenMP版	-lasl64 -mkl=parallel
ASLSTAT	32bit整数版	-laslstat -mkl=sequential
	64bit整数版	-laslstat64 -mkl=sequential
ASLQUAD	32bit整数/逐次版	-laslquad
	32bit整数/OpenMP版	-laslquad
Lis	逐次版	-llis_seq
	OpenMP版	-llis_omp
ARPACK		-larpack -mkl=sequential
XBLAS		-lxblas
METIS		-lmetis

表 13. NumericFactory のリンクオプション（逐次版/OpenMP 版 C プログラムから利用する場合）

ライブラリ名	リンクオプション	
ASL	32bit整数/逐次版	-laslcint -lasl -mkl=sequential -lifcore -limf
	64bit整数/逐次版	-laslcint64 -lasl64 -mkl=sequential -lifcore -limf lilp64
	32bit整数/OpenMP版	-laslcint -lasl -mkl=parallel
	64bit整数/OpenMP版	-laslcint64 -lasl64 -mkl=parallel
ASLSTAT	32bit整数版	-laslstatc -laslstat -mkl=sequential -lifcore -limf
	64bit整数版	-laslstatc64 -laslstat64 -mkl=sequential -lifcore -limf -lilp64
ASLQUAD	32bit整数/逐次版	-laslquadc++ -laslquad -lifcore -limf
	32bit整数/OpenMP版	-laslquadc++ -laslquad -lifcore -limf
dSFMT		-ldsfmt
SFMT		-lsfmt
SuperLU		-lsuperlu -mkl=sequential
Lis	逐次版	-llis_seq
	OpenMP版	-llis_omp
XBLAS		-lxblas
METIS		-lmetis

表 14. モジュール/ヘッダファイル（逐次版/OpenMP 版）

ライブラリ名	使用する言語	モジュール/ヘッダファイル
ASL	Fortran	不要
	C	asl.h
ASLSTAT	Fortran	不要
	C	aslstat.h
ASLQUAD	Fortran	aslquad.mod
	C++	aslquad.h
dSFMT	C	dSFMT.h
SFMT	C	SFMT.h
SuperLU	C	slu_sdefs.h (単精度実数版) slu_ddefs.h (倍精度実数版) slu_cdefs.h (単精度複素数版) slu_zdefs.h (倍精度複素数版)
Lis	Fortran	lisf.h
	C	lis.h
ARPACK	Fortran	不要
XBLAS	Fortran	不要
	C	blas_extended.h
METIS	C	metis.h

● MPI 版プログラムの場合

各ライブラリのリンクには、mpiifort、mpiicc コマンドを使用します。利用するプログラム言語に応じて、リスト 19、20 のようにリンクしてください。リンクオプションは使用するライブラリと言語に応じて指定します（表 15、表 16）。

NumericFactory でサポートしているライブラリを使用する場合、ライブラリによってはユーザプログラム側でモジュールファイルやヘッダファイルをインクルードする必要があります（表 17）。

Fortran から ASL または ASLSTAT の 64 ビット整数に対応したライブラリ利用する場合、コンパイル時に必ずオプション"-i8"を付けてコンパイルしてください。このオプションは、integer 型を 64 ビット整数と翻訳する Intel コンパイラのオプションです。

リスト 19. Fortran の場合（MPI 版）

```
[front1 ~]$ mpiifort source.f90  <リンクオプション>
```

リスト 20. C の場合（MPI 版）

```
[front1 ~]$ mpiicc source.c  <リンクオプション>
```

表 15. NumericFactory のリンクオプション（MPI 版 Fortran プログラムから利用する場合）

ライブラリ名		リンクオプション
ASL	32bit整数/逐次版	-lasl -mkl=sequential
	64bit整数/逐次版	-lasl64 -mkl=sequential
	32bit整数/OpenMP版	-lasl -mkl=parallel
	64bit整数/OpenMP版	-lasl64 -mkl=parallel
ASLSTAT	32bit整数版	-laslstat -mkl=sequential
	64bit整数版	-laslstat64 -mkl=sequential
ASLQUAD	32bit整数/逐次版	-laslquad
	32bit整数/OpenMP版	-laslquad
MUMPS	単精度実数版	-lsmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential
	倍精度実数版	-ldmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential
	単精度複素数版	-lcmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential
	倍精度複素数版	-lzmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential
Lis	逐次版	-llis_seq
	OpenMP版	-llis_omp
	MPI版	-llis_mpi
	MPI版+OpenMP版	-llis_omp_mpi
ARPACK		-larpack -mkl=sequential
PARPACK	MPI版	-lparpack -larpack -mkl=sequential
	BLACS版	-lparpack-blacs -larpack -lmkl_blacs_intelmpi_lp64 -mkl=sequential
XBLAS		-lxblas
METIS		-lmetis

表 16. NumericFactory のリンクオプション（MPI 版 C プログラムから利用する場合）

ライブラリ名		リンクオプション
ASL	32bit整数/逐次版	-laslcint -lasl -mkl=sequential -lifcore -limf
	64bit整数/逐次版	-laslcint64 -lasl64 -mkl=sequential -lifcore -limf -lilp64
	32bit整数/OpenMP版	-laslcint -lasl -mkl=parallel
	64bit整数/OpenMP版	-laslcint64 -lasl64 -mkl=parallel
ASLSTAT	32bit整数版	-laslstatc -laslstat -mkl=sequential -lifcore -limf
	64bit整数版	-laslstatc64 -laslstat64 -mkl=sequential -lifcore -limf -lilp64
ASLQUAD	32bit整数/逐次版	-laslquadc++ -laslquad -lifcore -limf
	32bit整数/OpenMP版	-laslquadc++ -laslquad -lifcore -limf
dSFMT		-ldsfmt
SFMT		-lsfmt
SuperLU		-lsuperlu -mkl=sequential
MUMPS	単精度実数版	-lsmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential -lifcore -limf
	倍精度実数版	-ldmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential -lifcore -limf
	単精度複素数版	-lcmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential -lifcore -limf
	倍精度複素数版	-lzmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -mkl=sequential -lifcore -limf
Lis	逐次版	-llis_seq
	OpenMP版	-llis_omp
	MPI版	-llis_mpi
	MPI版+OpenMP版	-llis_omp_mpi
XBLAS		-lxblas
METIS		-lmetis

表 17. モジュール/ヘッダファイル（MPI 版）

ライブラリ名	使用する言語	モジュール/ヘッダファイル
ASL	Fortran	不要
	C	asl.h
ASLSTAT	Fortran	不要
	C	aslstat.h
ASLQUAD	Fortran	aslquad.mod
	C++	aslquad.h
dSFMT	C	dSFMT.h
SFMT	C	SFMT.h
SuperLU	C	slu_sdefs.h (単精度実数版) slu_ddefs.h (倍精度実数版) slu_cdefs.h (単精度複素数版) slu_zdefs.h (倍精度複素数版)
MUMPS	Fortran	smumps_struc.h (単精度実数版) dmumps_struc.h (倍精度実数版) cmumps_struc.h (単精度複素数版) zmumps_struc.h (倍精度複素数版)
	C	smumps_c.h (単精度実数版) dmumps_c.h (倍精度実数版) cmumps_c.h (単精度複素数版) zmumps_c.h (倍精度複素数版)
Lis	Fortran	lisf.h
	C	lis.h
ARPACK PARPACK	Fortran	不要
	C	不要
XBLAS	Fortran	不要
	C	blas_extended.h
METIS	C	metis.h

■ Intel 製ライブラリ

【機能概要】

表 18 で示したライブラリが利用可能です。

表 18. Intel 製ライブラリの機能概要

ライブラリ名	機能概要
数値演算ライブラリ MKL (Math Kernel Library)	工学、科学、金融向けの数値演算関数を提供する。最適化とマルチスレッド化されたライブラリです。
画像処理ライブラリ IPP (Integrated Performance Primitives)	マルチメディア、データ処理、通信／信号処理などのアプリケーションを作成するための、最適化された基関数から構成されるライブラリです。
マルチスレッドライブラリ TBB (Threading Building Blocks)	アプリケーションをマルチスレッド化する場合に最適な C++テンプレート・ライブラリです。

【ライブラリのリンク方法】

- MKL

以下の Intel Math Kernel Libraly リンクアドバイザーをご利用ください。Select Intel Product では Intel MKL 11.1 を選択してください。

http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

- IPP, TBB

各ライブラリのマニュアルをご覧ください。

■　マニュアル

■ NEC NumericFactory

ライブラリのマニュアルを並列コンピュータ上で提供しています。front.cc.tohoku.ac.jp にログインし、以下のディレクトリから閲覧してください。

```
/usr/ap/NFMAN200
```

■ Intel コンパイラ、Intel 製ライブラリ

コンパイラと各ライブラリのマニュアルを並列コンピュータ上で提供しています。front.cc.tohoku.ac.jp にログインし、以下のディレクトリから閲覧してください。

```
/opt/intel/composerxe/Documentation
```

7 章 プログラムについての補足

■ プログラムの使用メモリサイズ

プログラムを実行した際、使用するメモリサイズをバイト単位で表示します(リスト 21)。あらかじめ、必要とするメモリサイズが判断できます。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。

【形式】　　**size**　実行形式ファイル名

リスト 21. 使用メモリサイズの表示

```
front1$ size a.out
1046912 + 140272 + 418928 = 1606112        1,606,112 バイト使用します
```

■ バイナリファイルの扱い(Fortran の場合)

センター以外のマシン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。センターでは、並列コンピュータ LX 406Re-2 のエンディアン仕様は Big-Endian に設定しています。Little-Endian のバイナリファイルを扱う場合は、環境変数 F_UFMTENDIAN の設定をクリアします。設定はホームディレクトリの.chsrc やバッチリクエストファイルに記述します。

Little-Endian 仕様のファイルを扱う設定(csh 形式)

【形式】　　unsetenv F_UFMTENDIAN

■ メモリ使用量が 2GB を越える配列を扱う方法

コンパイルオプションに"-mcmodel=medium"または"-mcmodel=large"の指定とともに"-shared-intel"を指定してください。

・-mcmodel=medium　コードは IP 相対アドレス指定、データは絶対アドレス指定でアクセスされます
・mcmodel=large　コードもデータも絶対アドレス指定でアクセスされます

メモリ使用量が 2GB を越える配列を扱う場合のコンパイル方法

【形式】　　ifort -mcmodel=large -shared-intel　オプション ソースファイル名

■ アプリケーションプログラム

表 19 は、センターでサービスを行うアプリケーションプログラム一覧です。それぞれの詳しい利用方法は、以下の Web ページまたは本誌 85 ページの「アプリケーションサービスの紹介」をご参照ください。

http://www.ss.cc.tohoku.ac.jp/application/index.html

表 19. アプリケーションソフトウェアとサービスホスト

アプリケーションソフトウェア		サービスホスト
分子軌道計算ソフトウェア	Gaussian	front.cc.tohoku.ac.jp
反応経路自動探索プログラム	GRRM11	
統合型数値計算ソフトウェア	Mathematica	
汎用構造解析プログラム	Marc/Mentat	
対話型解析ソフトウェア	MATLAB	

8 章　利用負担金

■　利用負担金について

利用負担金は、演算負担経費、ファイル負担経費、出力負担経費、可視化負担経費の 4 つがあります（表 20、表 21）。スーパーコンピュータと並列コンピュータを利用すると、演算負担経費が発生します。

共有利用は利用するノード数と経過時間によって負担額が決定し、計算資源を利用者間で共有利用する利用形態です。

また、占有利用は計算資源を待ち時間なく占有して利用することができ、申請する利用期間によって負担額が決定する利用形態です。

請求書は四半期（3 ヶ月）ごとに、利用者を取りまとめている支払い責任者に発行します。最新の情報は以下の Web サイトをご覧ください。

```
http://www.ss.cc.tohoku.ac.jp/utilize/academic.html   （学術利用）
http://www.ss.cc.tohoku.ac.jp/utilize/business.html   （民間期間利用）
```

表 20.　基本利用負担金（大学・学術利用）

区　分	項　目	利用形態	負　担　額
演　算 負担経費	スーパー コンピュータ	共有	利用ノード数 1　(実行数、実行時間の制限有)　無料(備考2)
			利用ノード数 1～32 まで　経過時間 1 秒につき　0.06 円
			利用ノード数 33～256 まで　経過時間 1 秒につき　(利用ノード数-32)×0.002 円+0.06 円
			利用ノード数 257 以上　経過時間 1 秒につき　(利用ノード数-256)×0.0016 円+0.508 円
		占有	利用ノード数 32　利用期間 3 ヶ月につき　400,000 円
			利用期間 6 ヶ月につき　720,000 円
			利用ノード数 64　利用期間 3 ヶ月につき　720,000 円
			利用期間 6 ヶ月につき　1,300,000 円
			利用ノード数 128　利用期間 3 ヶ月につき　1,300,000 円
			利用期間 6 ヶ月につき　2,340,000 円
	並列 コンピュータ	共有	利用ノード数 1～6 まで　経過時間 1 秒につき　0.04 円
			利用ノード数 7～12 まで　経過時間 1 秒につき　0.07 円
			利用ノード数 13～18 まで　経過時間 1 秒につき　0.1 円
			利用ノード数 19～24 まで　経過時間 1 秒につき　0.13 円
		占有	利用ノード数 1　利用期間 3 ヶ月につき　160,000 円 (可視化システムの 20 時間無料利用を含む) 利用期間 6 ヶ月につき　320,000 円 (可視化システムの 40 時間無料利用を含む)
ファイル 負担経費	1TB まで無料、追加容量 1TB につき年額		3,000 円
出力 負担経費	大判プリンタによるカラープリント		フォト光沢用紙 1 枚につき　600 円 クロス 1 枚につき　1,200 円
可視化 機器室利用 負担経費	1 時間の利用につき		2,500 円

21.　基本利用負担金（民間機関利用）

区　分	項　目	利用形態	負　担　額
演　算 負担経費	スーパー コンピュータ	共有	利用ノード数 1　(実行数、実行時間の制限有)　無料(備考2)
			利用ノード数 1～32 まで　経過時間 1 秒につき　0.18 円
			利用ノード数 33～256 まで　経過時間 1 秒につき　(利用ノード数-32)×0.006 円+0.18 円
			利用ノード数 257 以上　経過時間 1 秒につき　(利用ノード数-256)×0.0048 円+1.524 円

		占有	利用ノード数 32	利用期間 3 ヶ月につき	1,200,000 円
				利用期間 6 ヶ月につき	2,160,000 円
			利用ノード数 64	利用期間 3 ヶ月につき	2,160,000 円
				利用期間 6 ヶ月につき	3,900,000 円
			利用ノード数 128	利用期間 3 ヶ月につき	3,900,000 円
				利用期間 6 ヶ月につき	7,020,000 円
	並列コンピュータ	共有	利用ノード数 1〜6 まで	経過時間 1 秒につき	0.12 円
			利用ノード数 7〜12 まで	経過時間 1 秒につき	0.21 円
			利用ノード数 13〜18 まで	経過時間 1 秒につき	0.3 円
			利用ノード数 19〜24 まで	経過時間 1 秒につき	0.39 円
		占有	利用ノード数 1	利用期間 3 ヶ月につき (可視化システムの 20 時間無料利用を含む)	480,000 円
				利用期間 6 ヶ月につき (可視化システムの 40 時間無料利用を含む)	960,000 円
ファイル負担経費	1TB まで無料、追加容量 1TB につき年額				9,000 円
出力負担経費	大判プリンタによるカラープリント			フォト光沢用紙 1 枚につき	1,800 円
				クロス 1 枚につき	3,600 円
可視化機器室利用負担経費	1 時間の利用につき				7,500 円

備考

1　負担額算定の基礎となる測定数量に端数が出た場合は、切り上げる。
2　負担額が無料となるのは専用のジョブクラスで実行されたものとし、制限時間を超えた場合には強制終了する。
3　占有利用期間は年度を超えないものとし、期間中に障害、メンテナンス作業が発生した場合においても、原則利用期間の延長はしない。また、占有利用期間中のファイル負担経費は 10TB まで無料とする。
4　ファイル負担経費については申請日から当該年度末までの料金とする。

■　利用負担金の確認方法

● プロジェクトコードの確認（project コマンド）

利用負担金はプロジェクトコード毎に合算されます。利用可能なプロジェクトコードの確認、デフォルトのプロジェクトコードの設定は project コマンドをご利用ください。プロジェクトコードの名称変更、追加などについては共同利用支援係までお問い合わせください。

リスト 22.　プロジェクトコードの確認

```
front1 $ project
使用可能なプロジェクトおよびキュー名は次のとおりです

利用者番号：（利用者番号）
```

```
----------------------  使用可能なプロジェクト一覧  ----------------------

  プロジェクト名称      :  運営費交付金
  プロジェクトコード  :  un0000
  使用可能なキュー名  :  sx32 sx64 ・・・

デフォルトのジョブ実行プロジェクトは un0000 です

    1.デフォルトプロジェクト変更      9.終了

何番の処理を選びますか ？
```

● プロジェクト課金情報表示（pkakin コマンド、ukakin コマンド）

コマンドを実行した利用者が利用可能なプロジェクトについて、負担額や請求情報を表示します。負担額は前日の午前 9 時までに終了したリクエストの利用額までが反映されています。

リスト 23. プロジェクトごとの負担額、請求情報の表示

```
front1 $ pkakin
   2 月 20 日  現在の利用負担金は次のとおりです

プロジェクトコード       累計負担額       調整額累計       固定費合計          請求済額   今期請求予定額（うち請求持越額）
un0000               15,404             0             0             0       15,404               0

支払責任者と経理担当者の所属（学校、学部）が異なる場合はセンターに連絡してください
支払費目の指定は各部局の経理担当者（学内の場合）、もしくはセンター会計係（学外の場合）へお伝えください

    1.負担金の明細表示   9.終了

何番の処理を選びますか ? 1
出力する年度を入力して下さい （ 1.今年度   2.前年度 ） : 1
開始月を入力してください : 2
終了月を入力してください : 2
出力先を選択してください  （ 1.画面   2.ファイル ） : 1

================================================================================
                                プロジェクト負担金明細情報

    2 月 20 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
  支払責任者番号      :  aaaaaa
```

```
  プロジェクト名称      :  運営費交付金
  プロジェクトコード    :  un0000

                        合計          演算 SX          演算 LX         ファイル           出力          可視化
  利用額              15,143          14,635              508              0              0              0
  負担額              15,143          14,635              508              0              0              0
-------------------------------------------------------------------------------------------------------------
月別利用額
         2 月             15,143
-------------------------------------------------------------------------------------------------------------
利用者別利用額
        abc000               21
        abc001                6
           :                  :
        abc020               27
        abc021            4,895
```

リスト 24. プロジェクトごとの利用額情報の表示

```
front1 $ ukakin

        利用者番号=（利用者番号）

 ----------------------  利 用 額  ----------------------
  支払責任者          :  東北　太郎
  支払責任者番号      :  aaaaaa
  プロジェクト名称    :  運営費交付金
  プロジェクトコード  :  un0000

   月       演算 SX       演算 LX       ファイル          出力          可視化          合計
   4              0             0              0             0              0             0
   5              0             0              0             0              0             0
   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            170            10              0             0              0           180
   1            546           162              0             0              0           708
   2              0             0              0             0              0             0
   3              0             0              0             0              0             0
 合計           716           172              0             0              0           888
```

● ジャーナル（利用明細）の抜粋（ulist コマンド、plist コマンド）

コマンドを実行した利用者のジャーナル情報、プロジェクトごとのジャーナル情報を抜粋して CSV 形式（カンマ区切り）のファイルに出力します。表 22 の項目が出力されます。

リスト 25. 利用者のジャーナル情報を抜粋

```
front1 $ ulist

出力する年度を入力して下さい ( 1.今年度  2.前年度 ) : 1
開始月を入力してください : 2
終了月を入力してください : 2
出力するファイル名を入力してください ( 省略時 : ulist.csv ) :

ホスト ID は次のとおりです
  20:SX  04:LX  02:front  05:ストレージ  06:プリンタ  08:可視化
```

リスト 26. プロジェクトのジャーナル情報を抜粋

```
front1 $ plist

出力する年度を入力して下さい ( 1.今年度  2.前年度 ) : 1
開始月を入力してください : 2
終了月を入力してください : 2
出力するファイル名を入力してください ( 省略時 : plist.csv ) :
プロジェクトを選択してください

    1.(un0000)      運営費交付金

何番のプロジェクトを選びますか ? 1

ホスト ID は次のとおりです
  20:SX  04:LX  02:front  05:ストレージ  06:プリンタ  08:可視化
```

表 22. ulist コマンドの出力項目

課金計上年月,ジャーナル連番,利用者番号,支払責任者番号,プロジェクトコード,ホスト ID,クラス ID,キュー名,投入日時,開始日時,終了日時,経過時間,使用ノード数,ユーザ CPU 時間,最大メモリサイズ,ベクトル時間,ベクトル演算率,ノード時間（使用量）,利用額

● ジャーナルの集計（usum コマンド、psum コマンド）

コマンドを実行した利用者のジャーナル情報、プロジェクトごとのジャーナル情報を集計して表示します。

リスト 27. 利用者のジャーナル情報を集計

```
front1 $ usum

出力する年度を入力して下さい ( 1.今年度  2.前年度 ) : 1
開始月を入力してください : 2
終了月を入力してください : 2

----------------------  利 用 情 報  ----------------------

利用者番号  プロジェクトコード  ホスト ID       経過時間合計  ユーザ CPU 時間合計     最大メモリサイズ  ノード時間(使用量)合計
abc001      un0000              02        1850:24:22          :  : 0             997            2:14:39
abc001      un0000              20         150:34:21       4141:51:07          59,649         4090:20:56
ホスト ID は次のとおりです
  20:SX  04:LX  02:front  05:ストレージ  06:プリンタ  08:可視化
```

リスト 28. プロジェクトのジャーナル情報を集計

```
front1 $ psum

出力する年度を入力して下さい ( 1.今年度  2.前年度 ) : 1
開始月を入力してください : 2
終了月を入力してください : 2
プロジェクトを選択してください

    1.(un0000)     運営費交付金

何番のプロジェクトを選びますか ? 1

----------------------  利 用 情 報  ----------------------

プロジェクトコード  ホスト ID       経過時間合計  ユーザ CPU 時間合計     最大メモリサイズ  ノード時間(使用量)合計
un0000              02         14354:22:54          :  : 0          33,856            9:51:13
un0000              04            25:16:44       238:45:30         118,672           47:59:41
un0000              05               :  : 0          :  : 0               0              :  : 1
un0000              06               :  : 0          :  : 0               0              :  :15
un0000              20           247:49:04       5930:43:04          61,414         6711:14:15

ホスト ID は次のとおりです
  20:SX  04:LX  02:front  05:ストレージ  06:プリンタ  08:可視化
```

■ 利用見込み額の設定

プロジェクトコードの請求先が学外の場合は、請求処理の都合上 2 月中旬以降、3 月末までの利用額を見込み金額として設定します。設定した金額は、1 月から 2 月中旬までの利用負担金に合算し、2 月末に請求いたします。見込み金額の設定は mikomi コマンドで行います。

リスト 29.　mikomi コマンドの例

```
front1 $ mikomi

プロジェクトを選択してください
プロジェクトコード : プロジェクト名称　支払責任者番号 : 支払責任者氏名
un0000        : 運営費交付金        aaaaaa      : 東北　太郎

プロジェクトコードを入力してください un0000

 2 月 1 日 現在の見込み額は次のとおりです

  支払責任者      : 東北　太郎
  支払責任者番号 : aaaaaa
  プロジェクト名称 : 運営費交付金
  プロジェクトコード: un0000

  見込み額              :         0 円
  今期請求予定額        :   300,000 円
  合計請求額(見込み額込) :   300,000 円

 1.見込み額の指定 2.削除 9.終了
 何番の処理を選びますか ? 1

  見込み額を入力してください (円) ? 200000

  登録してよろしいですか (yes/no) ? yes

 2 月 1 日 現在の見込み額は次のとおりです

  支払責任者      : 東北　太郎
  支払責任者番号 : aaaaaa
  プロジェクト名称 : 運営費交付金
  プロジェクトコード: un0000
  見込み額指定者 : 東北　太郎 (利用者番号 変更日: 2 月 1 日)

  見込み額              :     200,000 円
  今期請求予定額        :     500,000 円
  合計請求額(見込み額込) :     500,000 円

 見込み額の指定 2.削除 9.終了
 何番の処理を選びますか ?
```

9章　運用に関する情報、お問い合わせ

■　運用に関するお知らせ

大規模科学計算システムのホームページで利用方法やシステムの運用状況について、最新情報をお知らせします。計画停電による運用の停止スケジュールなどについても、こちらでお知らせしております。

URL http://www.ss.cc.tohoku.ac.jp/

■　利用申請

利用申請に関する詳細は、以下の Web ページをご覧ください。申請書の提出、お問合せは共同利用支援係までお願いいたします。

URL http://www.ss.cc.tohoku.ac.jp/utilize/index.html
メールアドレス　uketuke@cc.tohoku.ac.jp
電話番号　022-795-3406

■ ストレージの追加手順、申請

ホームディレクトリの追加容量:1TB につき年額 3,000 円です。追加容量の申請は、以下のページより「ストレージ【追加／削減】申請書」をダウンロードして必要事項を記入の上、郵送またはメールでお送りください。

URL http://www.ss.cc.tohoku.ac.jp/utilize/form.html

送付先(共同利用支援係)

住所　〒980-8578　宮城県仙台市青葉区荒巻字青葉６番３号　東北大学サイバーサイエンスセンターセンター・本館
メールアドレス　uketuke@cc.tohoku.ac.jp

利用に関するお問い合わせ

■ システムの利用方法についてのお問合せ

利用相談員が対応いたします。以下のメールアドレスまでお問合せください。

メールアドレス sodan05@cc.tohoku.ac.jp

■ 利用負担金についてのお問合せ

共同利用支援係までお問い合わせください。

メールアドレス uketuke@cc.tohoku.ac.jp
電話番号 022-795-6251

■ 請求書についてのお問合せ

請求書の発送等については会計係までお問い合わせください。

メールアドレス kaikei@cc.tohoku.ac.jp
電話番号 022-795-3405

10 章　おわりに

ジョブ管理システム NQSⅡを中心に利用方法の解説と、ライブラリの紹介をしました。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、お気軽にセンターまでお問い合わせください。