［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

はじめに

並列コンピュータ LX 406Re-2 は、最大 576 コアによる並列処理や、ベクトル化に不向きなプログラムの高速な演算が可能です。また、Gaussian 等のアプリケーションプログラムは、高速ディスクアクセスが可能な SSD ドライブを搭載する専用ノードで実行されます。バッチ処理はジョブ管理システム NQSⅡを使用しています。この資料では、LX 406Re-2 の利用法について説明します。

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。

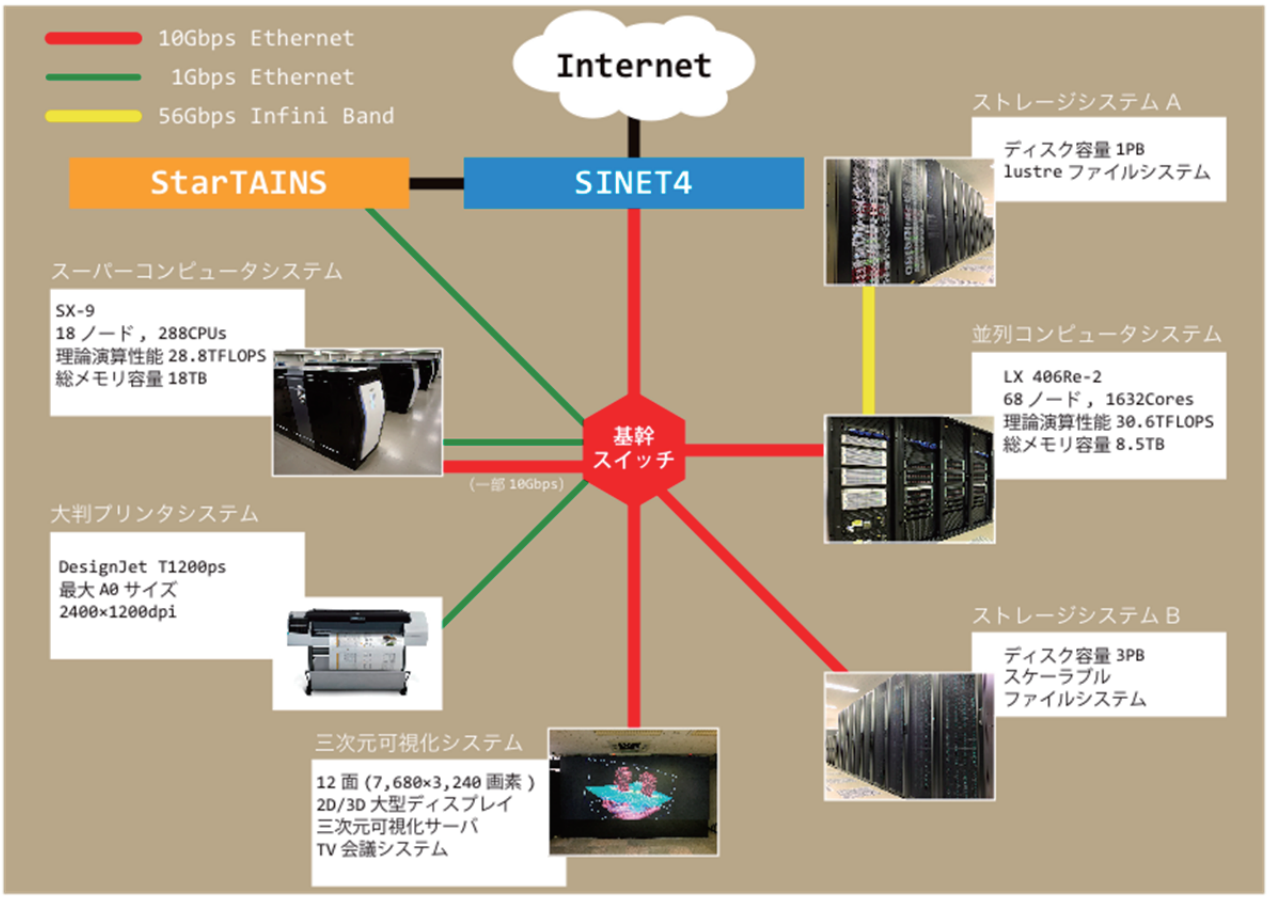

図 1. 大規模科学計算システム構成図

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-9 のフロントエンドサーバとしての役割も担っています。

スーパーコンピュータ SX-9

スーパーコンピュータ SX-9 は 1 ノードに、16 個の CPU と 1TB の主記憶装置を搭載し、合計 18 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 16 並列まで可能で、ノードあたりの最大演算性能は 1.6TFLOPS（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 64 並列まで実行可能です。

ログイン

並列コンピュータにログインします。ログインは、SSH(Secure SHell)プログラムを利用します（リスト1）。接続ホスト名は front.isc.tohoku.ac.jp です。
はじめて接続する場合はコマンド入力後に
”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  front.isc.tohoku.ac.jp -l 利用者番号
Password: パスワード

[利用者番号@front1 ~]$
```

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

センターに利用登録すると、図1すべてのシステムが利用可能となります。ログイン名とパスワードは各システムで共通です。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、定期的な変更をお願いします。公開鍵暗号方式によるログインも可能です。

利用登録時のログインシェルは csh を設定しています。bash 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

```
front1$ yppasswd
Changing password for 利用者番号.
Changing password 利用者番号
(current)UNIX Password: 現在のパスワード
New UNIX password: 新しいパスワード
Retype new UNIX password: 新しいパスワード（再度）
Password: all authentication tokens updated successfully.
```

リスト3 ログインシェルの変更（bash への変更例）

```
front1$ ypchsh
Changing NIS account information for 利用者番号 on front.
Please enter Password: パスワード

Changing login shell for 利用者番号 on front.
To accept the default, simply press return.
Login Shell [/bin/csh]: /bin/bash
The login shell has been changed on kanri-ux.
```

・ 並列コンピュータの日本語環境は UTF-8 です。日本語を表示させる場合には、接続ソフトウェアの文字コードを UTF-8 としてください。

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.isc.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.isc.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 14 ページの**マニュアル**をご覧ください。

■ Fortran

【形式】　**ifort**　オプション　Fortran ソースファイル名

　　mpiifort　オプション　Fortran ソースファイル名（MPI）

主なオプション

-parallel	自動並列化機能を利用する。
-par-report	自動並列化されたループの行番号を表示する。
-openmp	OpenMP を利用する。
-openmp-report	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++

【形式】 **icc オプション C ソースファイル名**
icpc オプション C++ソースファイル名

mpiicc オプション C ソースファイル名 （MPI）
mpiicpc オプション C++ソースファイル名 （MPI）

主なオプション

-parallel	自動並列化機能を利用する。
-par-report	自動並列化されたループの行番号を表示する。
-openmp	OpenMP を利用する。
-openmp-report	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++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。

ライブラリ

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 2）。

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 3）。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 2. 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 3. 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 4、5 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 4、表 5）。

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 6）。

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 4. Fortran の場合（逐次版/OpenMP 版）

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

リスト 5. C の場合（逐次版/OpenMP 版）

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

表 4. 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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	64bit 整数/逐次版	-lasl64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core
	32bit 整数/OpenMP 版	-lasl -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
	64bit 整数/OpenMP 版	-lasl64 -lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core
ASLSTAT	32bit 整数版	-laslstat -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	64bit 整数版	-laslstat64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core
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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core	
XBLAS	-lxblas	
METIS	-lmetis	

表 5. 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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	64bit 整数/逐次版	-laslcint64 -lasl64 -lmkl_intel_ilp64 -lmkl_sequential –pthread -lmkl_core -lifcore -limf
	32bit 整数/OpenMP 版	-laslcint -lasl -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -lifcore -limf
	64bit 整数/OpenMP 版	-laslcint64 -lasl64 -lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core -lifcore -limf
ASLSTAT	32bit 整数版	-laslstatc -laslstat -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	64bit 整数版	-laslstatc64 -laslstat64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core - pthread -lifcore -limf

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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread	
Lis	逐次版	-llis_seq
	OpenMP 版	-llis_omp
XBLAS	-lxblas	
METIS	-lmetis	

表 6. 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 6、7 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 7、表 8）。

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 9）。

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 6. Fortran の場合（MPI 版）

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

リスト 7. C の場合（MPI 版）

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

表 7. 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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	64bit 整数/逐次版	-lasl64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -ilp64
	32bit 整数/OpenMP 版	-lasl -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
	64bit 整数/OpenMP 版	-lasl64 -lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core -ilp64
ASLSTAT	32bit 整数版	-laslstat -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	64bit 整数版	-laslstat64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -ilp64
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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	倍精度実数版	-ldmumps -lmumps_common -lpord -lmetis

	倍精度実数版	-ldmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	単精度複素数版	-lcmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	倍精度複素数版	-lzmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
Lis	逐次版	-llis_seq
	OpenMP 版	-llis_omp
	MPI 版	-llis_mpi
	MPI 版+OpenMP 版	-llis_omp_mpi
ARPACK	-larpack -lmkl_intel_lp64 -lmkl_sequential -lmkl_core	
PARPACK	MPI 版	-lparpack -larpack -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
	BLACS 版	-lparpack-blacs -larpack -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core
XBLAS	-lxblas	
METIS	-lmetis	

表 8. 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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	64bit 整数/逐次版	-laslcint64 -lasl64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lifcore -limf -ilp64
	32bit 整数/OpenMP 版	-laslcint -lasl -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -lifcore -limf
	64bit 整数/OpenMP 版	-laslcint64 -lasl64 -lmkl_intel_ilp64 -lmkl_intel_thread

		-lmkl_core -pthread -lifcore -limf -ilp64
ASLSTAT	32bit 整数版	-laslstatc -laslstat -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	64bit 整数版	-laslstatc64 -laslstat64 -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf -ilp64
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 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core	
MUMPS	単精度実数版	-lsmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	倍精度実数版	-ldmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	単精度複素数版	-lcmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -lifcore -limf
	倍精度複素数版	-lzmumps -lmumps_common -lpord -lmetis -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -pthread -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 9. 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 10 で示したライブラリが利用可能です。

表 10. 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.isc.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.isc.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
```

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、ジョブの実行手続きを書いたシェルスクリプト(バッチリクエスト)を作成します。作成したファイルを NQSⅡに投入することで、ジョブの実行依頼をします。NQSⅡでは並列数やメモリサイズの違いにより複数のジョブクラスを設定しています。プログラムに合わせ適切なジョブクラスを選択し、そのキュー名を指定してバッチリクエストを投入します。バッチリクエストの実行順番が来ると、NQSⅡが自動的にジョブを実行します。

バッチリクエストの投入後は、リクエストの状態確認や、キューの混雑状況の確認、投入済みのリクエストをキャンセルすることが可能です。プログラムが終了するとリクエストはキュー情報から消え、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。

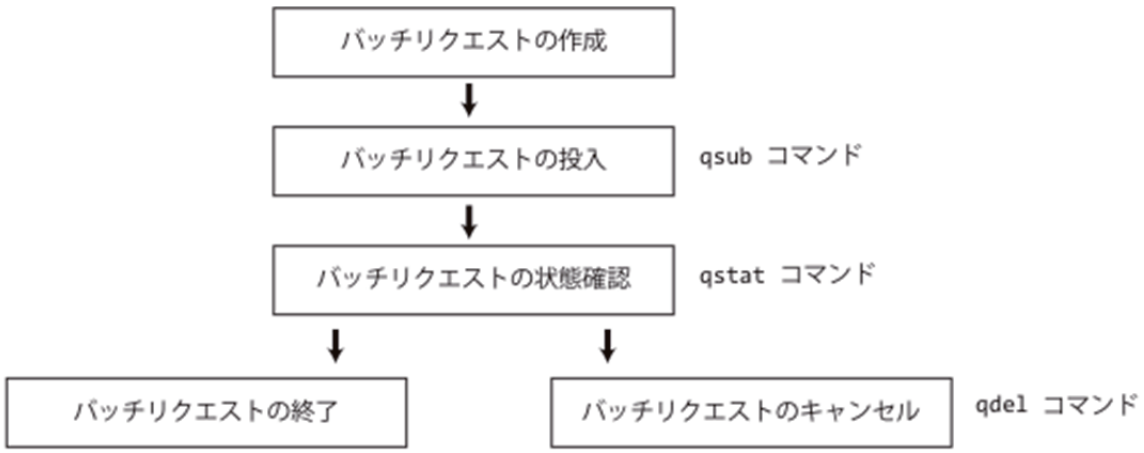

図 2 NQSII によるリクエストの流れ

【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 8 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 8. 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（リスト 9）。

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 9. 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 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 10）。

正しい指定方法:　　setenv FORT2 datafile

リスト 10. 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 FORT1 datafile          装置番号 1 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に投入して行います。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 11 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 11.　qsub コマンドの実行例

```
front1$ qsub -q n6 job-a
Request 1234.job submitted to queue: n6.
```

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@front.isc.tohoku.ac.jp」宛に送られます。

● **キュー名**

-q オプションで指定するキュー名の一覧です(表 11)。並列化されていないプログラムは ns キューに、自動並列/OpenMP により並列化されているプログラムは nh, n1 どちらかのキューに、MPI により並列化されているプログラムは nh, n1, n6, n12, n24 のいずれかのキューに投入します。プログラムの並列数に応じて適

切なキューにリクエストを投入してください。2 ノード以上を利用した並列実行には MPI の利用が必用です。mg はアプリケーション専用のキューです。

表 11.　並列コンピュータのジョブクラス

キュー名	利用ノード数 （コア数）	時間制限 ［時間］	メモリ容量 ［GB］
ns	1 (1)	無制限	5
nh	1 (24)	1（経過時間）	128
n1	1 (24)	無制限	128
n6	6 (144)	〃	128×6
n12	12 (288)	〃	128×12
n24	24 (576)	〃	128×24
mg（ｱﾌﾟﾘｹｰｼｮﾝ専用）	1 (24)	〃	128

● 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 12 は、2 行目で n6 のキュー名を指定(-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オプションの方を有効とします。

リスト 12.　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 n6                      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（リスト 13）。

【形式】　　qstat

リスト 13. qstat コマンド表示例

```
front1$ qstat
RequestID      ReqName  UserName  Queue   Pri STT  S   Memory  ACCPU Elapse  R H M Jobs
-------------- -------- -------- -------  --- ---  - --------  ----  ------  - - - ----
  370.job      reqname  利用者番号   n6     0  RUN  -  732.1B  42350   43012  Y Y Y    1
1:371.job      jobA     利用者番号  n12     0  QUE  -  0.0B        0      0   Y Y Y    1
1:373.job      jobB     利用者番号  n24     0  QUE  -  0.0B        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 14）。

リスト 14. ジョブの終了

```
front1$ 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。各キュー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(リスト15)。

【形式】　　qstat　-Q

リスト 15.　-Q オプションの表示例

```
front1$ 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
-------------- --- ---- ------- ---- ----------------------------------------------------------------
ns              0    0 ENA ACT   32    1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
nh              0    0 ENA ACT   32    1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
n1              0    0 ENA ACT   32    1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
n6              0    0 ENA ACT   32    1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
n12             0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
n24             0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
mg              0    0 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
p16             1    4 ENA ACT   32    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
p32             1    2 ENA ACT   32    0   0   0   0   0   0   0   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   0
s               0    1 ENA ACT   32    0   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   0
-------------- --- ---- ------- ---- ----------------------------------------------------------------
 <TOTAL>                           1   0   0   0   0   1   0   0   0   0   0   0   0   0   0
-------------- --- ---- ------- ---- ----------------------------------------------------------------
```

・枠で囲った箇所が、並列コンピュータのキューです。

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 16. qdel コマンドの表示例

```
front1$ qdel 1234
Request 1234.job was deleted.
```

■ 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 17）。

-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。

リスト 17.　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                    コメント行
#PBS -q n6                     埋め込みオプション（n6 キューに投入）
cd work                        作業ディレクトリへ移動
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 18）。

リスト 18. 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                    コメント行
#PBS -q n6                     埋め込みオプション（n6 キューに投入）
setenv OMP_NUM_THREADS 24      自動並列/Open MP での並列数
cd work                        作業ディレクトリへ移動
mpirun -ppn 1 -np 6 a.out      MPI プログラムの実行
```

■ 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 19）。表 12 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 19. 会話型処理の例（a.out を実行する）

```
yourhost$ ssh front.isc.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 12. 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

その他

■ 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 20）。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 20. 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 13 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、センターの Web ページをご覧ください。

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

表 13. 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.isc.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	

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。