

**統計数理研究所**  
**統計科学スーパーコンピュータシステム**  
**ベクトル計算機利用の手引**

**第 1.1 版**

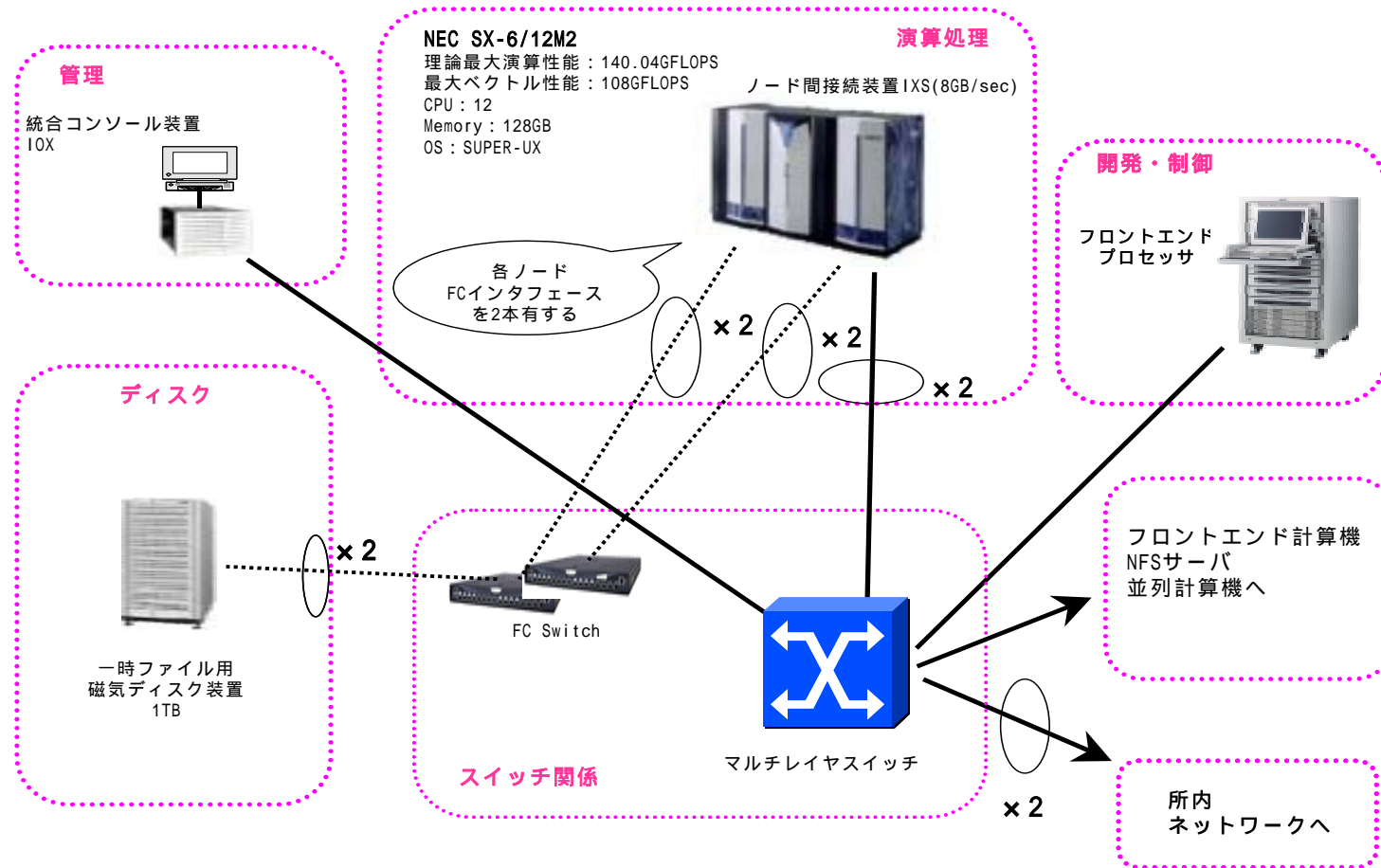


## 目次

1. システム構成.....	1
1.1. システム構成図.....	1
1.2. IP アドレス一覧.....	2
1.3. ソフトウェア構成.....	2
1.3.1. ベクトル計算機.....	2
1.3.2. ベクトル計算機フロントエンドサーバ.....	3
2. システム利用形態.....	4
2.1. サーバの位置付け.....	4
2.2. システム利用イメージ.....	4
3. システム利用.....	5
3.1. ログイン方法.....	5
3.2. パスワード変更.....	6
3.3. ファイルシステム.....	7
4. 開発環境.....	8
4.1. ツール一覧.....	8
4.2. クロスコンパイラ.....	8
4.3. FORTRAN90/SX.....	10
4.3.1. クロスコンパイル.....	10
4.3.2. マニュアル.....	10
4.4. C++/SX.....	10
4.4.1. クロスコンパイル.....	10
4.4.2. マニュアル.....	11
4.5. MPI/SX.....	12
4.5.1. クロスコンパイル.....	12
4.5.2. 指定形式について.....	12
4.5.3. マニュアル.....	12
4.6. 各種情報の取得.....	13
4.7. 時間計測関数.....	16
4.7.1. Fortran 関数(clock, etime).....	16
4.7.2. C 関数(clock,syssex).....	17
5. プログラム実行方法.....	18
5.1. NQSII の概要.....	18
5.2. NQSII キュー.....	18
5.3. 本システム NQSII 構成.....	19
5.4. バッチリクエストの作成から終了まで.....	20
5.5. マルチノード MPI ジョブ用バッチリクエストの作成.....	24
5.6. マニュアル.....	25

# 1. システム構成

## 1.1. システム構成図



## 1.2. IP アドレス一覧

機器名	ホスト名	IP アドレス	用途
SX-6 #0	ismsx1	133.58.226.82	ジョブ実行
SX-6 #1	ismsx2	133.58.226.83	ジョブ実行
Express5800	ismsx	133.58.226.81	ユーザログイン, ジョブ投入

## 1.3. ソフトウェア構成

### 1.3.1. ベクトル計算機

プロダクト名	機能
SUPER-UX 関連	OS・SX 基本ソフトウェア (R14.1)
NQSII	UNIX 上でバッチ処理を行う
FORTRAN90/SX	Fortran コンパイラ
C++/SX	C および C++ コンパイラ
OpenMP Fortran	共有メモリ向け並列処理の Fortran からのアプリケーションインタフェース
OpenMP C/C++	共有メモリ向け並列処理の C および C++ からのアプリケーションインタフェース
MPI/SX	分散並列処理プログラミングを行うためのメッセージ通信ライブラリ
MPI2/SX	MPI2.0 の主要機能を提供する分散並列処理プログラミングを行うためのメッセージ通信ライブラリ
PDBX	SX 上で動作するシンボリックリンクデバッガ dbx の全機能に加えて, 並列処理プログラムのデバッグを可能にするデバッガ
PSUITE サーバ	PSUITE クライアントから利用者プログラムの実行, デバッグ制御のために呼び出される
Vampirertrace/SX	MPI プログラム実行トレース情報採取用ライブラリ
ASL/SX	科学技術計算ライブラリ
ASLSTAT/SX	科学技術計算ライブラリ統計機能
ASLCINT/SX	科学技術計算ライブラリ C 言語インターフェイス
MathKeisan	数値計算ライブラリ

### 1.3.2. ベクトル計算機フロントエンドサーバ

プロダクト名	機能
RedHat Enterprise Linux	OS (AS 2.1)
Cross-kit/SX	SX用のクロスツールおよびクロスリンカ用環境
Fortran90/SX クロスコンパイラ	フロントエンド上で Fortran90/SX と同じオブジェクトを生成するクロスコンパイラ
C++/SX クロスコンパイラ	フロントエンド上で C++/SX と同じオブジェクトを生成するクロスコンパイラ
OpenMP クロス Fortran	OpenMP Fortran のクロス環境を提供
OpenMP クロス C/C++	OpenMP C/C++のクロス環境を提供
MPI/SX クロス	MPI/SX のクロス環境を提供
MPI2/SX クロス	MPI2/SX のクロス環境を提供
PSUITE クライアント	プログラム生成から、翻訳・実行・デバッグおよびチューニングまでの一連の作業を支援する統合プログラム開発環境ツール
Vampiretrace/SX cross	SX クロス環境上に提供される MPI 用プログラム実行トレース情報採取用ライブラリ

## 2. システム利用形態

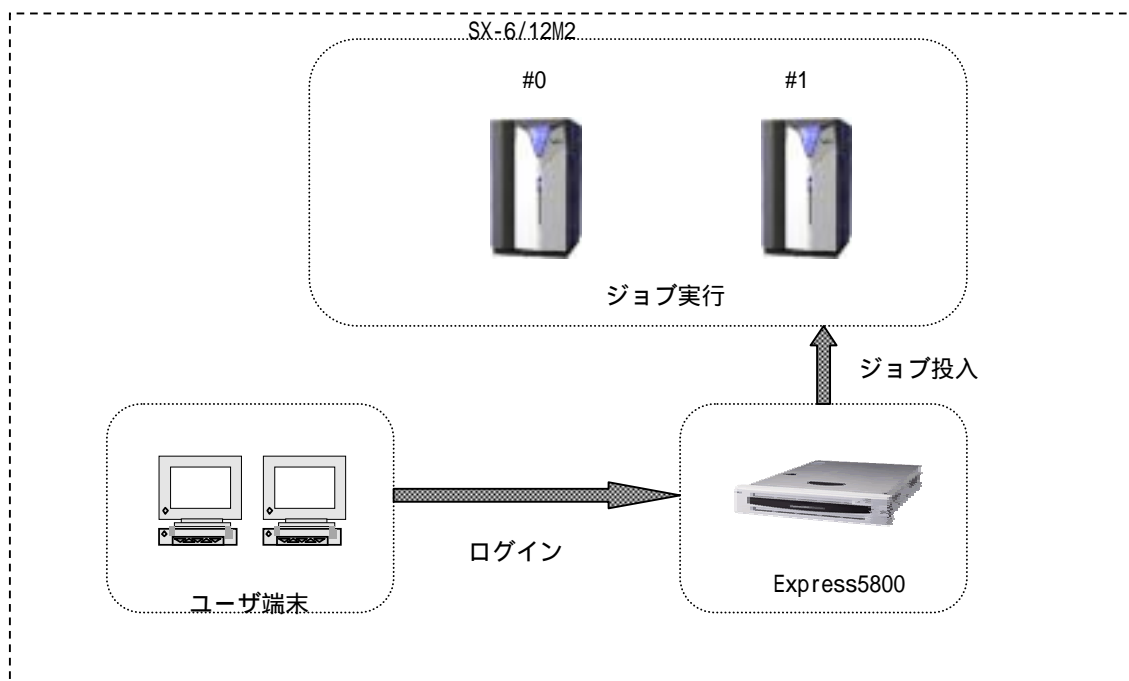
### 2.1. サーバの位置付け

以下に各サーバの位置付けを示す。

サーバ	CPU 数	メモリ	位置付け
ベクトル計算機 ismsx1, ismsx2 (SX-6/12M2)	12	128 GB	ベクトル型演算利用 ・大規模ベクトルジョブ実行 ・ノード内やノード間でのベクトル/並列演算 による超高速演算
フロントエンドサーバ ismsx.ism.ac.jp (Express5800)	8	4 GB	ユーザログインマシン プログラム開発, クロスコンパイル ジョブ制御

### 2.2. システム利用イメージ

利用者はフロントエンドサーバ ismsx.ism.ac.jp にログインし、プログラム開発、バッチジョブの  
。以下にシステム利用イメージを示す。



### 3. システム利用

#### 3.1. ログイン方法

```
$ telnet ismsx.ism.ac.jp
Trying...
Connected to ismsx.
Escape character is '^]'.
Local flow control on
Telnet TERMINAL-SPEED option ON
Red Hat Linux Advanced Server release 2.1AS/i686 (Pensacola)
login: nec                ユーザ名(この場合 nec)を入力する
Password:                 パスワードを入力(画面には表示されない)
Last login: Wed Jan  7 19:55:05 from XXX.XXX.XXX.XXX
[nec@ismsx nec]$
```

### 3.2. パスワード変更

本システムでは、フロントエンド上のユーザアカウント、パスワードは、NISにて管理しています。  
パスワードを変更する場合には `yppasswd` コマンドを使用します。

```
[nec@ismsx nec]$ yppasswd
Changing NIS account information for nec on ismsx.
Please enter old password: ██████████          旧パスワードを入力
Changing NIS password for nec on ismsx.
Please enter new password: ██████████         新パスワードを入力
Please retype new password: ██████████       新パスワードを再入力

The NIS password has been changed on ismsx.
                                パスワードが変更された旨メッセージが出力される
```

パスワードは、以下の字を組み合わせた6文字から8文字にしてください。

- 小文字のアルファベット
- 大文字のアルファベット
- 0から9の数字
- , (カンマ)

ただし、以下は禁止です。

- 数字だけのパスワード
- ユーザのログイン名と同じ
- ユーザのログイン名を反転したりずらしただけのパスワード
- 新しいパスワードと古いパスワードが3文字以上同じ
- 単語(固有名詞)
- 組み合わせの簡単なパスワード

### 3.3. ファイルシステム

本システムで一般ユーザが利用可能なファイルシステムは下表の通りです。

ismsx.ism.ac.jp(Express5800)

名称	マウントポイント	利用目的・備考
HOME 領域 (NFS)	/home0, /home1	<ul style="list-style-type: none"> <li>・ログイン時の初期化ファイル等を格納する。</li> <li>・ファイルサーバのディスクを NFS マウントした領域</li> </ul>
WORK 領域 (NFS)	/short	<ul style="list-style-type: none"> <li>・ジョブ実行時に使用する比較的大きいファイルを格納する。</li> <li>・保存期間は無期限</li> <li>・SX-6#0 のディスクを NFS マウントした領域</li> </ul>
	/tmpsx	<ul style="list-style-type: none"> <li>・ジョブ実行時に使用する比較的大きいファイルを格納する。</li> <li>・保存期間は 10 日間</li> <li>・SX-6#0 のディスクを NFS マウントした領域</li> </ul>

ismsx1.ism.ac.jp(SX-6#0)

名称	マウントポイント	利用目的・備考
HOME 領域 (NFS)	/home0, /home1	<ul style="list-style-type: none"> <li>・ログイン時の初期化ファイル等を格納する。</li> <li>・ファイルサーバのディスクを NFS マウントした領域</li> </ul>
WORK 領域 (ローカルディスク)	/short	<ul style="list-style-type: none"> <li>・ジョブ実行時に使用する比較的大きいファイルを格納する。</li> <li>・保存期間は無期限</li> <li>・ローカルディスク</li> </ul>
	/tmpsx	<ul style="list-style-type: none"> <li>・ジョブ実行時に使用する比較的大きいファイルを格納する。</li> <li>・保存期間は 10 日間</li> <li>・ローカルディスク</li> </ul>

ismsx1.ism.ac.jp (SX-6#1)

名称	マウントポイント	利用目的・備考
HOME 領域 (NFS)	/home0, /home1	<ul style="list-style-type: none"> <li>・ログイン時の初期化ファイル等を格納する。</li> <li>・ファイルサーバのディスクを NFS マウントした領域</li> </ul>
WORK 領域 (NFS)	/short	<ul style="list-style-type: none"> <li>・ジョブ実行時に使用する比較的大きいファイルを格納する。</li> <li>・保存期間は無期限</li> <li>・SX-6#0 のディスクを GFS マウントした領域</li> </ul>
	/tmpsx	<ul style="list-style-type: none"> <li>・ジョブ実行時に使用する比較的大きいファイルを格納する。</li> <li>・保存期間は 10 日間</li> <li>・SX-6#0 のディスクを GFS マウントした領域</li> </ul>

## 4. 開発環境

### 4.1. ツール一覧

本システムでは以下の環境が利用可能です。

	ismsx1.ismsx2 (SX-6#0, #1)	ismsx.ism.ac.jp (Express5800)
言語	Fortran90, C, C++ のコンパイル及び実行	Fortran90, C, C++ のクロスコンパイル
ライブラリ	MPI/SX, MPI2/SX ASL, MathKeisan	MPI/SX, MPI2/SX ASL, MathKeisan
エディタ	-	Vi

### 4.2. クロスコンパイラ

#### (1) クロスコンパイラに関して

クロスコンパイラとは、フロントエンドサーバ上で SX-6 のオブジェクトファイルを生成するソフトで、生成したモジュールは SX-6 上で実行可能です。

プログラムのコンパイル、リンクなどはフロントエンドマシン上で行い、SX-6 は主にプログラムの実行に利用した方がリソースの有効活用が図れます。

またコンパイル性能は、SX-6 上セルフコンパイラよりもフロントエンド上のクロスコンパイラの方が高速です。

## (2) コマンド

フロントエンド上で SX-6 用の FORTRAN90/SX および C++/SX の実行モジュールを作成(コンパイル/リンク)する場合、次のコマンドを使用します。

機能概要	コマンド名
FORTRAN90/SX クロスコンパイルコマンド	sxf90 SX-6 上の f90 コマンドに相当
C++/SX クロスコンパイルコマンド	sxc++ SX-6 上の c++コマンドに相当
SUPER-UX 用共通オブジェクト・ファイルのリンク・エディタ	sxld SX-6 上の ld コマンドに相当
SUPER-UX 用コンパイラ・コンパイラ	sxyacc SX-6 上の yacc コマンドに相当
SUPER-UX 用簡単な字句解析プログラムの作成	sxlex SX-6 上の lex コマンドに相当
SUPER-UX 用 C プログラム・チェッカ	sxlint SX-6 上の lint コマンドに相当
SUPER-UX 用オブジェクト・ファイルの部分ダンプ	sxdump SX-6 上の dump コマンドに相当
SUPER-UX 用オブジェクト・ファイルアーカイバ	sxar SX-6 上の ar コマンドに相当
SUPER-UX 用オブジェクトの名前リスト出力	sxnm SX-6 上の nm コマンドに相当
SUPER-UX 用オブジェクト・ライブラリの順序関係の検索	sxldorder SX-6 上の lorder コマンドに相当
SUPER-UX 用オブジェクト・ライブラリの注釈セクションの操作	sxmcs SX-6 上の mcs コマンドに相当
SUPER-UX 用オブジェクト・ライブラリのセクションのバイト数の出力	sxsize SX-6 上の size コマンドに相当
SUPER-UX 用オブジェクト・ライブラリのシンボルと行番号情報の削除	sxstrip SX-6 上の strip コマンドに相当

## (3) 利用方法

クロスコンパイル環境を利用する基本的な流れは次のようになります。

フロントエンド上にログインし、ホームディレクトリ等にコンパイルするソースを用意する。

クロスコンパイルコマンド(sxf90 あるいは sxc++)でクロスコンパイルを行う。

NQS ジョブとして SX-6 に投入する。

## (4) 注意事項

クロスコンパイラで作成されたモジュールはフロントエンド上では実行できません。SX-6 上で実行してください。

### 4.3. FORTRAN90/SX

FORTRAN90/SX はスーパーコンピュータ SX システムの能力を最大限に引き出すために、高度な自動ベクトル化機能ならびに最適化機能を備えた Fortran システムです。

フロントエンド上でのクロスコンパイルの使用を推奨します。

#### 4.3.1. クロスコンパイル

##### (1) 利用方法

フロントエンド上での FORTRAN90/SX によるコンパイルは以下のコマンドで行います。

```
[nec@ismsx nec]$ sxf90 (オプション) (プログラムファイル名)
```

##### (2) 利用例

Fortran プログラム sample.f をコンパイルし、実行ファイル sample.exe を生成する場合。

```
[nec@ismsx nec]$ sxf90 -o sample.exe sample.f
```

#### 4.3.2. マニュアル

以下のマニュアルを参照してください。

「FORTRAN90/SX 言語説明書」

「FORTRAN90/SX プログラミングの手引」

### 4.4. C++/SX

C++/SX はスーパーコンピュータ SX システムの能力を最大限に引き出すために、高度な自動ベクトル化機能ならびに最適化機能を備えた C++システムです。C++/SX コンパイラは C/SXv2 コンパイラとの互換性を有しますので、C プログラムのコンパイルにも C++コマンドを使用してください。

フロントエンド上でのクロスコンパイルの使用を推奨します。

#### 4.4.1. クロスコンパイル

##### (1) 利用方法

フロントエンド上での C++/SX によるコンパイルは以下のコマンドで行います。

```
[nec@ismsx nec]$ sxc++ (オプション) (プログラムファイル名)
```

##### (2) 利用例

C++プログラム sample.c をコンパイルし、実行ファイル sample.exe を生成する場合。

```
[nec@ismsx nec]$ sxc++ -o sample.exe sample.c
```

#### 4.4.2. マニュアル

以下のマニュアルを参照してください。

「C++/SX 言語説明書」

「C++/SX プログラミングの手引」

## 4.5. MPI/SX

MPI/SX および MPI2/SX はオペレーティングシステム SUPER-UX 上で分散並列処理プログラミングを行うためのメッセージ通信ライブラリです。SX アーキテクチャの特長の1つである共有メモリを活かした高速な通信を実現しています。

フロントエンド上でのクロスコンパイルの使用を推奨します。

### 4.5.1. クロスコンパイル

#### (1) 利用方法

フロントエンド上での MPI/SX をリンクする Fortran プログラムのコンパイルは以下のコマンドで行います。

```
[nec@ismsx nec]$ sxmpif90 (オプション) (プログラムファイル名)
```

#### (2) 利用例

Fortran プログラム sample.f をコンパイルし、実行ファイル sample.exe を生成する場合。

```
[nec@ismsx nec]$ sxmpif90 -o sample.exe sample.f
```

### 4.5.2. 指定形式について

MPI の実行プログラムを起動する場合、mpirun コマンドを用います。

mpirun による起動には、以下の2通りの方法があります。

- ・実行ノード名や実行プログラム名を mpirun に直接指示する（直接指定形式）
- ・実行定義ファイルに実行ノード名や実行プログラム名などを記述し、この実行定義ファイルを指定する（間接指定形式）

- ・直接指定形式

```
[nec@ismsx nec]$ mpirun (オプション) (実行プログラム名) [実行プログラムへ渡す引数]
```

- ・間接指定形式

```
[nec@ismsx nec]$ mpirun (オプション) -f (実行定義ファイル)
```

### 4.5.3. マニュアル

詳細は以下のマニュアルを参照して願います。

「MPI/SX 利用の手引き」

#### 4.6. 各種情報の取得

##### (1) 利用方法

SX-6 ではコンパイル時の翻訳リスト, コンパイラメッセージ, プログラム・インフォメーション, ファイル・インフォメーションをプログラムのコンパイルおよび実行時に取得することができます。

表示項目	表示内容
	実行方法
コンパイル時の 翻訳リスト, コンパイラメッセージ	プログラムのベクトル化情報等
	コンパイル時に オプション <code>-Wf ' -L fmtlist '</code> を指定する
Fortran プログラム インフォメーション	詳細な Fortran プログラム情報
	実行時に 環境変数 <code>" F_PROGINF "</code> を <code>" detail "</code> または <code>" DETAIL "</code> に設定する
C++ プログラム インフォメーション	詳細な C++ プログラム情報
	コンパイル時に オプション <code>-acct</code> を指定し, 実行時に 環境変数 <code>" C_PROGINF "</code> を <code>" detail "</code> または <code>" DETAIL "</code> に設定する
Fortran プログラム ファイルインフォメーション	FORTTRAN プログラムファイル I/O 性能情報
	実行時に 環境変数 <code>" F_FILEINF "</code> を <code>" yes "</code> または <code>" YES "</code> に設定する
C++ プログラム ファイルインフォメーション	C++ プログラムファイル I/O 性能情報
	コンパイル時に オプション <code>-acct</code> を指定し, 実行時に 環境変数 <code>" C_FILEINF "</code> を <code>" yes "</code> または <code>" YES "</code> に設定する

(2) 出力例

翻訳リスト出力例

```

:
:
100:          call basis(nx, nr, mx)
101:          id1 = nx
102: +-----> do 30 l = 1, mx
103: |          id0 = id1
104: |          id1 = id1/nr(l)
105: |          id2 = nx/nr(l)
106: |W-----> do 10 k = 1, nx
107: ||*-----> do 10 i = 1, nad2
108: |||          br(i,k) = 0.0d0
109: |||          bi(i,k) = 0.0d0
110: |W*----- 10 continue
111: |          klmx = nx/id0
112: |+-----> do 22 kl = 0, klmx-1
113: ||+-----> do 21 kl0 = 0, nr(l)-1
114: |||          call snfct(id1, kl, id2, kl0, nr(l), nx, fc, wr, wi)
115: |||          jk1 = id1*kl + id2*kl0
116: |||+----> do 23 jmljl = 1, nr(l)*id1
117: ||||          jml = (jmljl-1)/id1 + 1
118: ||||          jl = mod(jmljl-1, id1) + 1
119: ||||          jk0 = id1*( nr(l)*kl + (jml-1) )
120: ||||V--> do 15 i1 = 1, nad2
121: |||||          br(i1,jl+jk1) = br(i1,jl+jk1)
122: |||||          &          + wr(jml)*ar(i1,jl+jk0) - wi(jml)*ai(i1,jl+jk0)
123: |||||          bi(i1,jl+jk1) = bi(i1,jl+jk1)
124: |||||          &          + wr(jml)*ai(i1,jl+jk0) + wi(jml)*ar(i1,jl+jk0)
125: ||||V-- 15 continue
126: |||+---- 23 continue
127: ||+----- 21 continue
128: |+----- 22 continue
129: |W-----> do 27 i2 = 1, nx
130: ||*-----> do 27 i1 = 1, nad2
131: |||          ar(i1,i2) = br(i1,i2)
:
:
```

プログラムインフォメーション出力例

*****	プログラム	情報	*****
経過時間 (秒)	:		0.977348
ユーザ時間 (秒)	:		0.972147
システム時間 (秒)	:		0.003464
ベクトル命令実行時間 (秒)	:		0.964496
全命令実行数	:		24886864.
ベクトル命令実行数	:		11251402.
ベクトル命令実行要素数	:		2778786702.
浮動小数点データ実行要素数	:		1356298553.
MOPS 値	:		2872.427638
MFLOPS 値	:		1395.157759
平均ベクトル長	:		246.972484
ベクトル演算率 (%)	:		99.511698
メモリ使用量 (MB)	:		34.031250
MIPS 値	:		25.599896
命令キャッシュミス (秒)	:		0.001925
オペランドキャッシュミス (秒)	:		0.000263
バンクコンフリクト時間 (秒)	:		0.033023

## 4.7. 時間計測関数

### 4.7.1. Fortran 関数(clock, etime)

clock 関数、etime 関数をご紹介します。

#### (3) clock 関数

CPU 時間をマイクロ秒単位で返します。real\*8 型の値を返します。

```
real*8 time1  
call clock(time1)
```

例)

```
% cat ex6_1.f  
program ex6_1  
real*8 t1, t2  
call clock(t1)  
call sub()  
call clock(t2)  
print *, t2 t1  
end
```

#### (4) etime 関数

etime 関数は、プログラムの始めからの経過時間を（ユーザ時間、システム時間）をマイクロ秒単位で返します。

```
real*8 time1  
call etime(time1)
```

例)

```
% cat ex6_2.f  
program ex6_2  
real*8 t1, t2  
call etime(t1)  
call sub()  
call etime(t2)  
print *, t2 t1  
end
```

#### 4.7.2. C 関数(clock,sysx)

clock 関数、sysx 関数をご紹介します。

##### (5) clock 関数

clock は、clock への最初の呼び出しがあったときから使用された CPU 時間の長さ (マイクロ秒単位) を返します。クロック・チック単位での値が設定されるので CLOCKS\_PER\_SEC で割って時間に換算します。

```
#include <time.h>
clock_t clock(void);
```

例)

```
% cat ctime.c

#include <time.h>
#include <stdio.h>

int main(void)
{
    clock_t start, end;
    long l;
    start = clock();
    for (l=0; l<100000000; l++);
    end = clock();
    printf("The loop is :%f sec ¥n", (double)(end - start) / CLOCKS_PER_SEC);
    return 0;
}
```

##### (6) sysx 関数

sysx はマシン固有機能を実現します。cmd 引数が実行される機能を規定します。そのうちシステムの現在時刻(1970年1月1日00:00:00GMTからの経過時間)は、cmd が HGTIME で表示します。時刻は、マイクロ秒単位に変換します。

```
#include <sys/types.h>
#include <sys/sysx.h>
int sysx (int cmd, long long arg1, long long arg2,
          long long arg3, long long arg4, long long arg5);
```

例)

```
% cat etime.c

#include <sys/types.h>
#include <sys/sysx.h>
int main(void)
{
    unsigned long long start, end;
    long l,m;

    (void)sysx(HGTIME, &start);
    for (m = 1; m < 1000; m++)
        for (l=0; l<100000000; l++);
    (void)sysx(HGTIME, &end);
    printf("The loop is :%f sec ¥n",
           ((double)(end - start))/1000000);
    return 0;
}
```

## 5. プログラム実行方法

### 5.1. NQSII の概要

Network Queuing System II (NQSII) は、ハイ・パフォーマンス・クラスタシステム計算リソースを最大限活用するためのバッチ処理システムです。

NQSII は、従来の NQS の機能を包含し、ジョブキューイング機能、リソース管理機能などの主要機能をクラスタシステムに適応させる形で機能強化しています。

NQSII は、クラスタ単位にキュー、バッチリクエストを管理することによりシングルシステム環境 (SSE) を実現します。NQSII のクラスタシステム構成は、

- ・バッチリクエストを管理するバッチサーバホスト
- ・バッチジョブの実行を制御する実行ホスト
- ・ユーザコマンド、システムコンソール等を使って一般利用者がリクエスト操作を行うクライアントホスト

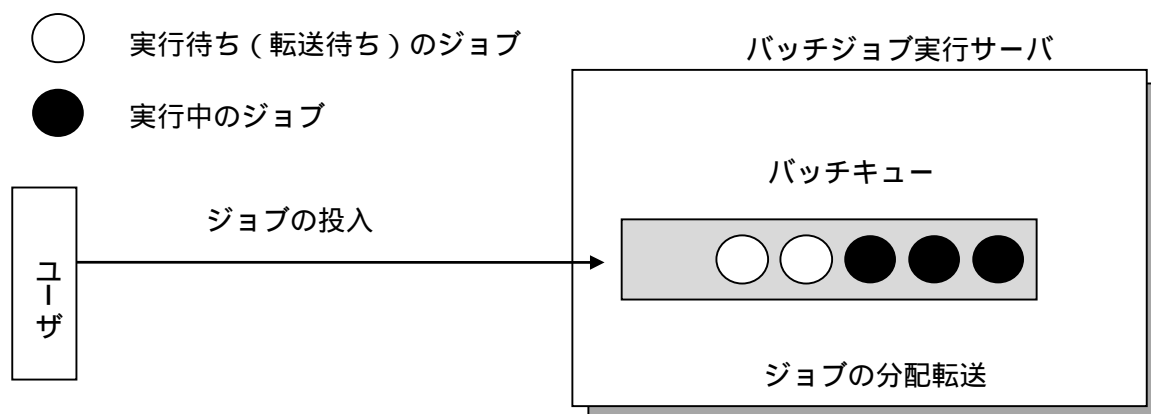
から構成されます。ただしこれは NQSII バッチシステムの各機能による論理的な区別であり、物理的には 1 台のマシン上で全てのホストを兼ねる事も、また、それぞれ異なったマシン上に各ホストを構成する事も可能です。

NQSII は、一括して処理するいくつかのコマンドを組み合わせたバッチリクエストや、排他的に利用する各種周辺装置に対するファイル出力リクエストを受け取り処理します。利用者はリクエストの投入、終了、監視、制限、管理を行うことができます。

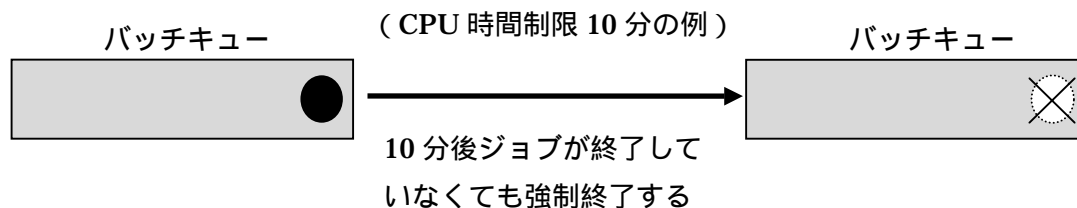
### 5.2. NQSII キュー

NQSII におけるキューとは、NQSII が受け付けたリクエストを一旦ためておくものであり、NQSII はこのキューに溜まっているリクエストを順番に実行していきます。

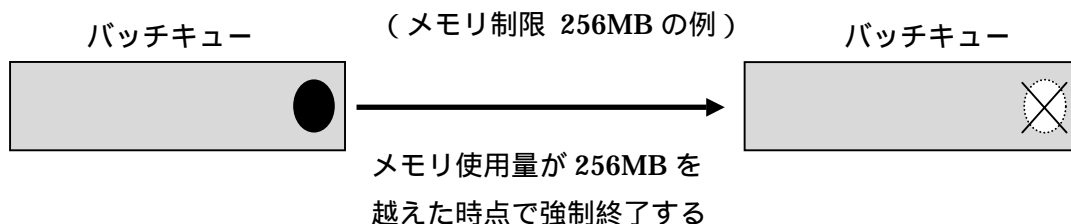
#### (1) キュー形態イメージ例



#### (2) CPU 時間制限をした場合

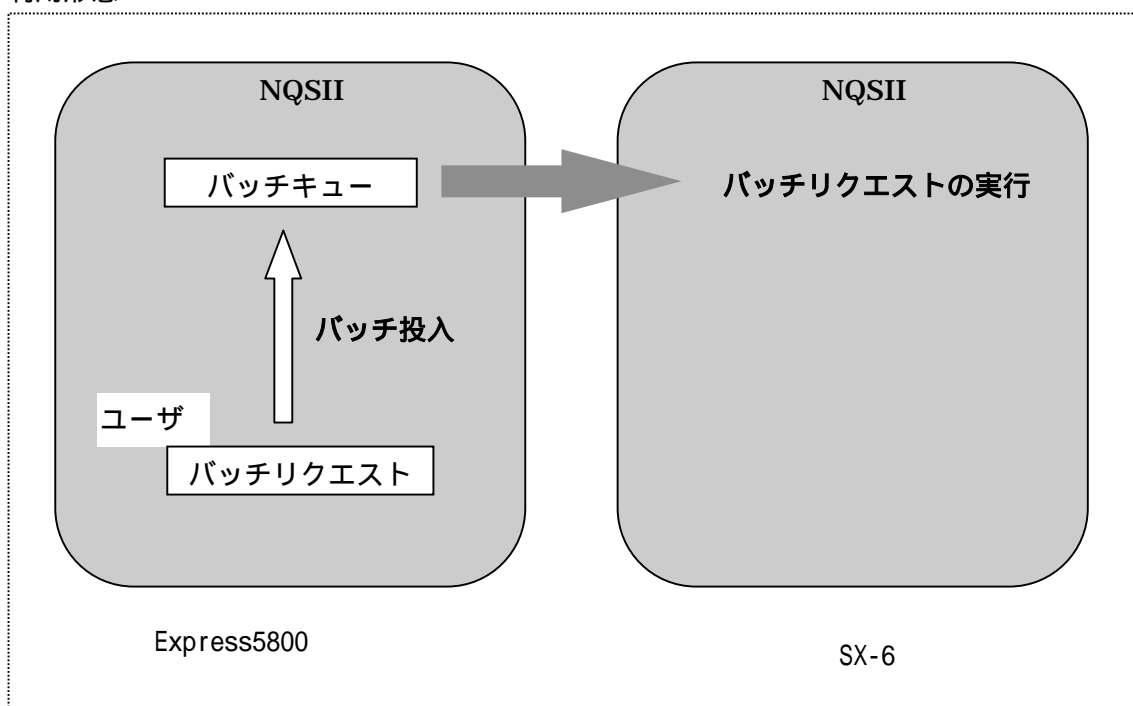


(3) メモリ制限をした場合



5.3. 本システム NQSII 構成

(1) 利用形態



(2) バッチキュー構成

バッチキューはバッチ処理を行うためのキューです。バッチ処理を行う場合には、バッチキューにバッチリクエストを投入する必要があります。本システムでは以下のバッチキューがあります。

	CPU 数	経過時間 (時)	メモリ (GB)		多重度	備考
			既定値	最大値		
q1	1	24	16	unlimit	4	シングルノード内での実行
q4	4	24	16	unlimit	2	シングルノード内での実行
q6	6	24	16	unlimit	1	シングルノード内での実行
q12	12	24	16	unlimit	1	ノード間での実行

## 5.4. バッチリクエストの作成から終了まで

以下にバッチリクエストの作成から終了までの手順を示します。

### (1) バッチリクエストの作成

NQSII では、一括して実行するシェルスクリプト・ファイルを作成する必要があります。以下にシェルスクリプト・ファイルの例を示します。

```
#!/bin/sh

# プログラムインフォメーション出力設定
F_PROGINF=detail
export F_PROGINF

# ジョブ実行ディレクトリへ移動
cd /home/nec

# プログラム実行
./testjobF >& testjobF.out
```

### (2) バッチリクエストの投入

NQSII では、作成したシェルスクリプトをバッチリクエストとして qsub コマンドによってホストシステムに投入します。本システムでは、フロントエンド上で qsub コマンドを実行します。qsub コマンド使用時には、投入キュー名を -q オプションにて指定してください。qsub コマンドのコマンド形式は以下のようになります。

```
qsub -q (投入キュー名) (オプション) (シェルスクリプトファイル名)
```

#### リクエスト投入時のオプション

qsub でリクエスト投入時に指定できる主なオプションについて説明します。

##### i) 結果ファイル関係オプション

```
-o [path/]file
```

リクエストの標準出力用結果ファイルを、path/で指定したディレクトリ配下の file という名で指定します。指定しない場合、リクエストを投入したディレクトリ配下に “リクエスト名.o.リクエスト連番” という名で標準出力用結果ファイルが作成されます。

```
-e [path/]file
```

リクエストの標準エラー出力用結果ファイルを、path/で指定したディレクトリ配下の file という名で指定します。指定しない場合、リクエストを投入したディレクトリ配下に “リクエスト名.e.リクエスト連番” という名で標準エラー出力用結果ファイルが作成されます。

ii) リクエスト名指定オプション

```
-N request-name
```

バッチリクエストの名前を指定します。 *name* は 63 文字まで指定できます。  
本オプションが指定されない場合、コマンド行で与えられたスクリプトファイルの名前がとられます。  
スクリプトを指定しない場合は、"STDIN" となります。

iii) ジョブ数の指定オプション

```
-b batch_job_count
```

実行されるバッチジョブ数を指定します。本オプションが指定されない場合、実行されるバッチジョブ数は、1 となります

iv) ジョブ数の指定オプション

```
-T batch_job_topology
```

実行されるバッチジョブの形態を指定します。 *batch\_job\_topology* には、次の指定ができます。

・ **distrib**

実行されるバッチジョブは分散ジョブです。 **-b** オプションで指定されたバッチジョブ数だけバッチジョブが生成され実行されます。このときのバッチジョブの内容はすべて同じ内容となります。

・ **mpisx**

実行されるバッチジョブは **mpisx** ジョブです。 **mpi** ジョブを実行する場合は、本オプションを指定する必要があります。

本オプションが指定されない場合、実行されるバッチジョブの形態は、**distrib** となります

v) 資源制限用オプション

```
-l resource_list[,resource_list,...]
```

リクエストやジョブ、プロセスに対する資源制限の最大値や警告値を設定します。  
*resource\_list* は以下のとおりです。制限の最大値と警告値の両方を指定する場合 " で囲む必要があります。

**elapstim\_req=["max\_limit",warn\_limit"]**

実行開始からの経過時間による制限(リクエスト)

**memsz\_job=["max\_limit",warn\_limit"]**

使用可能な最大メモリサイズの制限(ジョブ)

など。

### (3) バッチリクエストの状態確認

NQSII では、バッチリクエストに関する状態確認コマンドとして以下のものが利用可能です。

#### バッチリクエストの状態表示

投入したバッチリクエストの状態を確認するには、`qstat(1)` コマンドを用います。リクエストの指定はリクエスト ID で行いますので、リクエスト ID がわかっている場合はリクエストの直接指定が可能です。以下にその場合の例を示します。

```
[nec@ismsx nec]$ qstat 72
RequestID      ReqName  UserName Queue      Pri STT S   Memory      CPU R H Jobs
-----
72.ismsx.ism.ac  STDIN   nec      q1          0 RUN -   1.93M      0.23 Y Y   1
```

STT のカラムにリクエストの状態が表示されます。ここで表示される意味はそれぞれ以下のとおりです。

RUN	実行中
QUE	実行待ち状態
WAT	開始時刻の待ち合わせ中
HLD	保留中
HOL	保留要求中
SUS	実行一時中断の要求中、実行一時中断中、実行一時中断から実行再開中
RST	チェックポイントからのリスタート中
ARI	転送ブキューからの受信中
TRS	転送キューからの送信中
EXT	実行結果ファイルの転送中
PRR	スレーブリクエストの実行開始待ち状態 (マスタリクエストのみ)
POR	スレーブリクエストの実行終了待ち状態 (マスタリクエストのみ)
MIG	動的ジョブマイグレーションによるリクエスト移動中

STT 以外の項目の内容についてはマニュアルを参照してください。

リクエスト ID がわからないときは、現在登録されているすべてのバッチリクエストの情報を参照します。この場合は、`qstat(1)`コマンドでリクエスト ID を指定せずに実行します。

```
[nec@ismsx nec]$ qstat
RequestID      ReqName  UserName Queue      Pri STT S  Memory      CPU R H Jobs
-----
72. ismsx.ism.ac STDIN    nec     q1         0 RUN -   1.93M      0.23 Y Y   1
73. ismsx.ism.ac STDIN    nec     q1        20 QUE -   0.00B      0.00 Y Y   1
74. ismsx.ism.ac STDIN    nec     q2        20 QUE -   0.00B      0.00 N N   2
```

またリクエストの詳細情報が欲しい場合は、`qstat(1)`コマンドに**-f** オプションを付けて実行します

#### 実行キューの状態確認

キューの状態を確認するときは、`qstat(1)` コマンドで**-Q** オプションを用います。

`qstat(1)` コマンドにはシステム上の実行キューすべてを対象に情報を表示する機能(**-e** オプション)があるのでそれを使用します

```
[nec@ismsx nec]$ qstat -Q -e
[EXECUTION QUEUE] Batch Server Host: ismsx.ism.ac.jp
=====
QueueName      SCH JSVs ENA STS  PRI TOT ARR WAI QUE PRR RUN POR EXT HLD HOL RS
T SUS MIG STG CHK
-----
q1              0   2 ENA ACT  30  1  0  0  0  0  0  0  0  1  0  0
0  0  0  0  0
q12             0   2 ENA ACT  30  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0
q4              0   2 ENA ACT  30  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0
q6              0   2 ENA ACT  30  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0
-----
<TOTAL>                1  0  0  0  0  0  0  0  1  0  0
0  0  0  0  0
-----
```

ENA のカラムにはキューの第 1 特性、STS のカラムにはキューの第 2 特性、その後には各状態のリクエスト数などが表示されます

#### 第 1 特性

##### ENABLED 状態

キューはリクエストの登録を受け付ける状態です。

##### DISABLED 状態

キューはリクエストの登録を受け付けない状態です。

#### 第 2 特性

##### ACTIVE 状態

キューはリクエストの実行が可能な状態です。

##### INACTIVE 状態

キューはリクエストの実行が禁止されている状態です。

なお、実行キューの詳細情報が欲しい場合は、`qstat(1)` コマンドに `-f` オプションを付けて実行します

#### (4) バッチリクエストの終了

バッチリクエストの状態確認より、バッチリクエストが終了したことを確認します。また、バッチリクエストは任意の時点で強制的に終了させることが可能です。実行中のリクエストを終了させるときは `'-k'` を指定します。

Express5800 上から以下を実行します。

```
[nec@ismsx nec]$ qdel (リクエスト ID)
```

### 5.5. マルチノード MPI ジョブ用バッチリクエストの作成

例

キュー `q12` に MPI ジョブを投入する際のシェルスクリプト・ファイル (`go.sh`) の一例を示します。

```
#!/bin/sh
F_PROGINF=detail # プログラムインフォメーション出力設定
export F_PROGINF

# プログラム実行
mpirun -host 0 -np 6 -host 1 -np6 /home0/nec/prog/a.out
```

NQSII を利用して MPI プログラムを実行する場合、`-host` はジョブ番号を指定します。

このあと qsub コマンドでジョブを投入する際に -b オプションでジョブ数を 2 で指定するため、ジョブ番号は、それぞれ "0" と "1" を指定します。

シェルスクリプトファイルを作成したら以下のように qsub コマンドにてジョブを投入します。  
"-T mpisx" で MPI ジョブであることの指定、"-b 2" で SX-6 の各ノードでジョブを 1 つずつ実行させる指定、になります。

```
[nec@ismsx nec]$ cd /home0/nec/prog
[nec@ismsx nec]$ qsub -q q12 -T mpisx -b 2 ./go.sh
```

## 5.6. マニュアル

NQSII のコマンドに関する詳細は、フロントエンド上から以下のコマンドでご覧いただけます。

例) qsub コマンドのオンラインマニュアルを見る場合

```
[nec@ismsx nec]$ man -M /usr/man/nqsII qsub
```

また、NQSII で MPI プログラムを実行する場合の操作方法の詳細は以下のマニュアルをご参照下さい。

「MPI/SX 利用の手引き」第 3 章操作法      NQS ジョブにおける実行