

# サイバーメディアセンター 大規模計算機システムの利用

大阪大学 情報推進部 情報基盤課

# 本日のプログラム

- I. システムのご紹介
- II. 利用方法の解説・実習
  - i. システムへの接続
  - ii. プログラムの作成・コンパイル
  - iii. ジョブスクリプトの作成
  - iv. ジョブスクリプトの投入
- III. 利用を希望する方へ

# SX-ACE

## NEC製のベクトル型スーパーコンピュータ



|            | ノード毎          | 1クラスタ<br>(512ノード) | 総合<br>(3クラスタ) |
|------------|---------------|-------------------|---------------|
| CPU数       | 1             | 512               | 1536          |
| コア数        | 4             | 2048              | 6144          |
| 演算性能       | 276<br>GFLOPS | 141<br>TFLOPS     | 423<br>TFLOPS |
| ベクトル<br>性能 | 256<br>GFLOPS | 131<br>TFLOPS     | 393<br>TFLOPS |
| 主記憶容量      | 64GB          | 32TB              | 96TB          |

# VCC (大規模可視化対応PCクラスタ)

NEC製のスカラ型クラスタシステム  
GPU計算や可視化装置との連動が可能

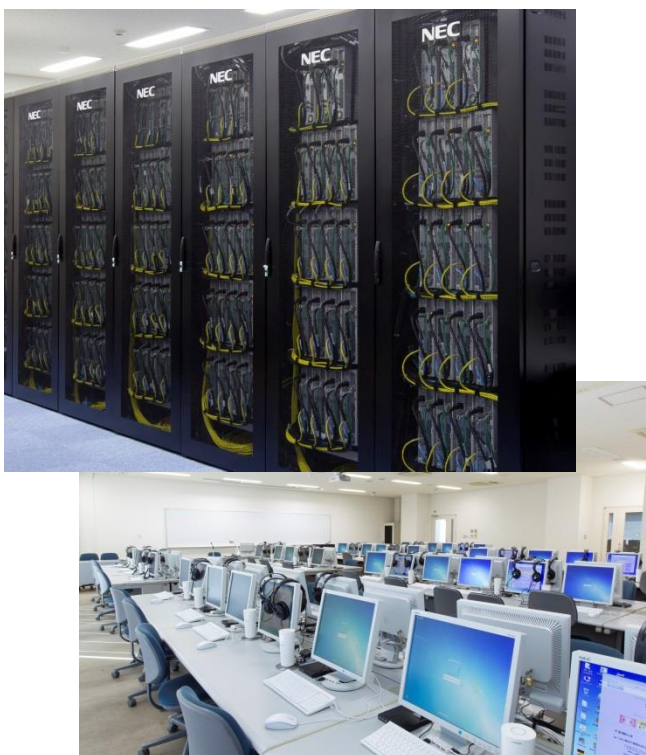


|           | 1ノード               | 総合<br>(65ノード)  |
|-----------|--------------------|----------------|
| CPU数      | 2                  | 130            |
| コア数       | 20                 | 1300           |
| 演算性能      | 400<br>GFlops      | 26.0<br>Tflops |
| 主記憶<br>容量 | 64 GB              | 4.160 TB       |
| GPU       | 59枚 (69.03 Tflops) |                |

# HCC (汎用コンクスタ)

NEC製のスカラ型クラスタシステム  
学生用の端末PCとしても使用している

2017年度  
更新予定



|           | 豊中地区           |               | 吹田地区           |               | 箕面地区           |               |
|-----------|----------------|---------------|----------------|---------------|----------------|---------------|
|           | 1<br>ノード       | 全体            | 1ノード           | 全体            | 1ノード           | 全体            |
| CPU数      | 2              | 536           | 2              | 338           | 2              | 276           |
| 演算性能      | 28.8<br>GFLOPS | 7.7<br>TFLOPS | 28.8<br>GFLOPS | 4.9<br>TFLOPS | 28.8<br>GFLOPS | 4.0<br>TFLOPS |
| 主記憶<br>容量 | 4GB            | 1.1TB         | 4GB<br>or 12GB | 1.2TB         | 4GB            | 0.6TB         |
| ノード数      | 268ノード         |               | 169ノード         |               | 138ノード         |               |
| 全ノード<br>数 | 575ノード         |               |                |               |                |               |

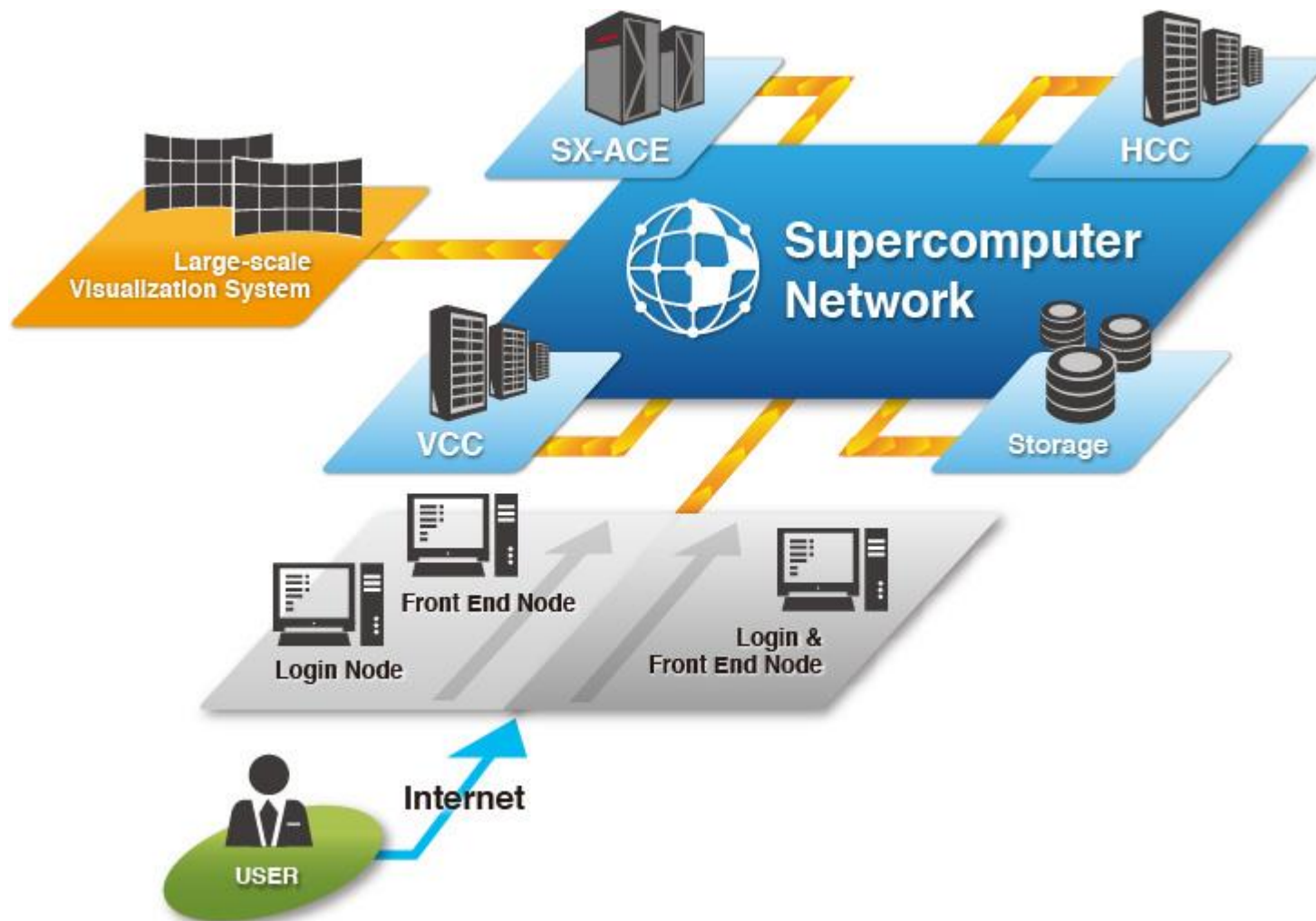
# フロントエンド端末

プログラムのコンパイルや計算結果の確認を行うための作業用端末

フロントエンド端末から各計算機に対して処理の実行を指示 ※詳細は後述

計算機自体へのログインは原則禁止(一部例外有)

# システム全体図



# 本日のプログラム

I. システムのご紹介

## II. 利用方法の解説・実習

- i. システムへの接続
- ii. プログラムの作成・コンパイル
- iii. ジョブスクリプトの作成
- iv. ジョブスクリプトの投入

III. 利用を希望する方へ



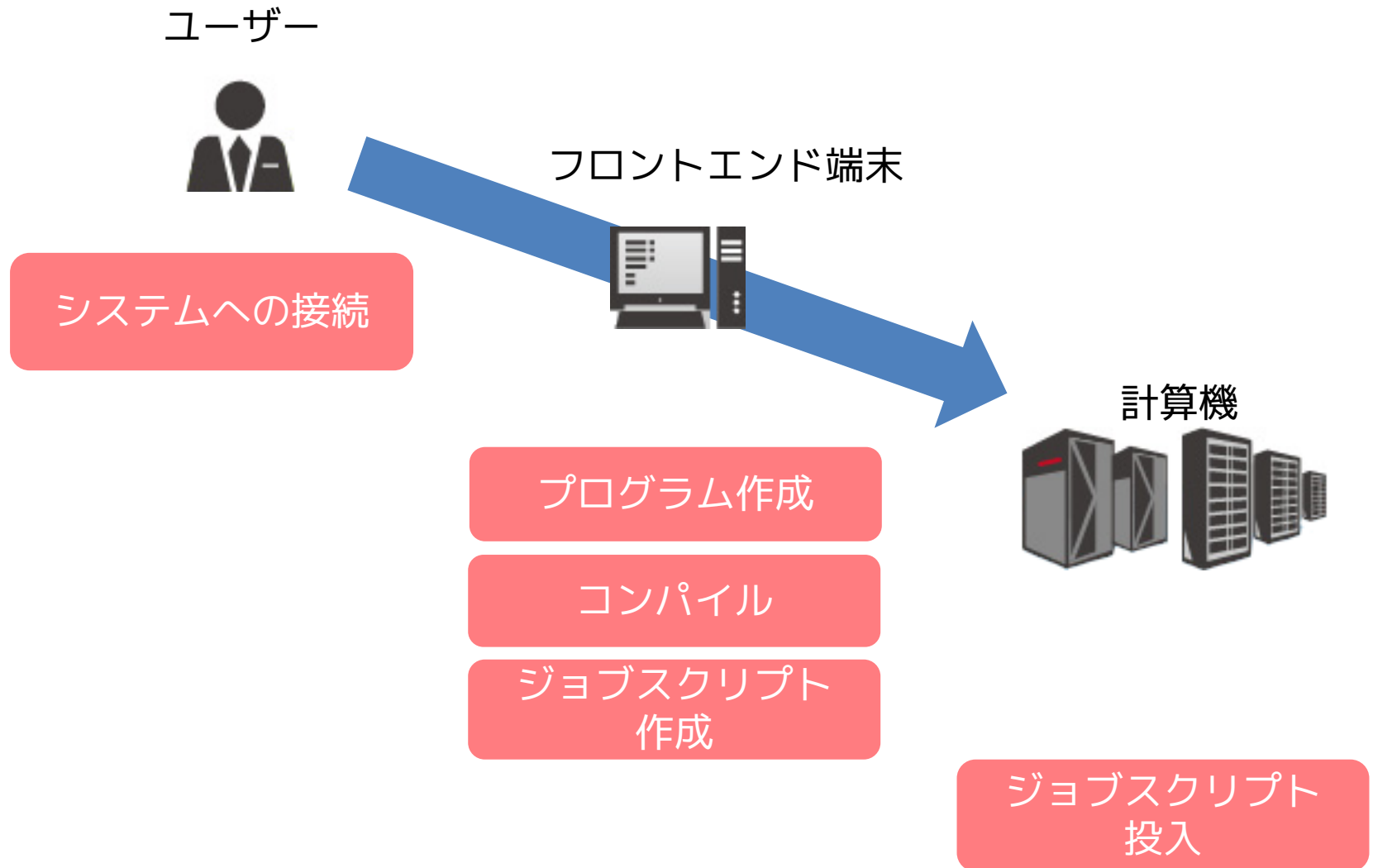
# 利用方法の解説・実習

本項では初心者を対象に  
大規模計算機システムの利用方法を解説します

途中、**実習も行います**

配布したアカウントは講習会後もしばらく利用可能  
ご自宅からでもシステムに接続できます

# 利用の流れ



# 本日のプログラム

I. システムのご紹介

## II. 利用方法の解説・実習

i. システムへの接続

ii. プログラムの作成・コンパイル

iii. ジョブスクリプトの作成

iv. ジョブスクリプトの投入

III. 利用を希望する方へ

# システムへの接続

ログインはSSH (Secure Shell) 接続

Win: TeraTermなど, Mac: ターミナルを使用

接続先は **login.hpc.cmc.osaka-u.ac.jp**

【参考】ファイル転送はSFTP接続

接続先は **ftp.hpc.cmc.osaka-u.ac.jp**

ユーザー



ログインサーバ



フロントエンド端末



SSH

login.hpc.cmc.osaka-u.ac.jp

4台あり、ログインユーザの  
少ない端末に自動で振り分ける

# 本日のプログラム

I. システムのご紹介

## II. 利用方法の解説・実習

i. システムへの接続

ii. プログラムの作成・コンパイル

iii. ジョブスクリプトの作成

iv. ジョブスクリプトの投入

III. 利用を希望する方へ

# プログラムの作成

計算機を利用するために、まずプログラムを作成する必要があります

今回はプログラムを用意しました

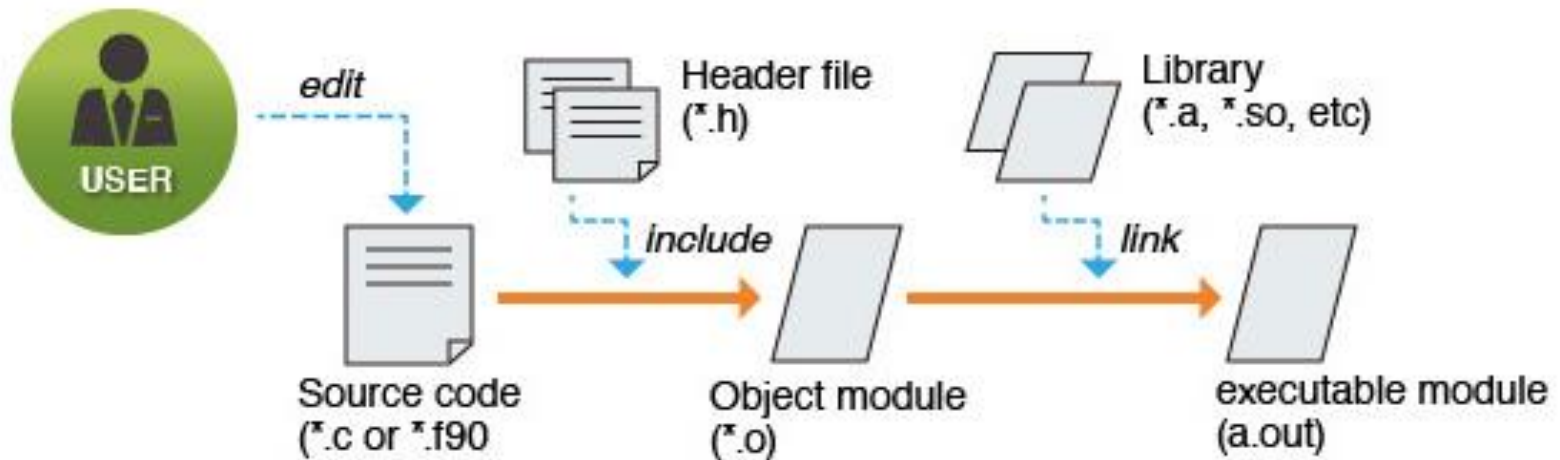
当センターの計算機で使用可能な言語

Fortran言語、C言語、C++言語

「プログラムの書き方」については  
特に説明しません

# コンパイル

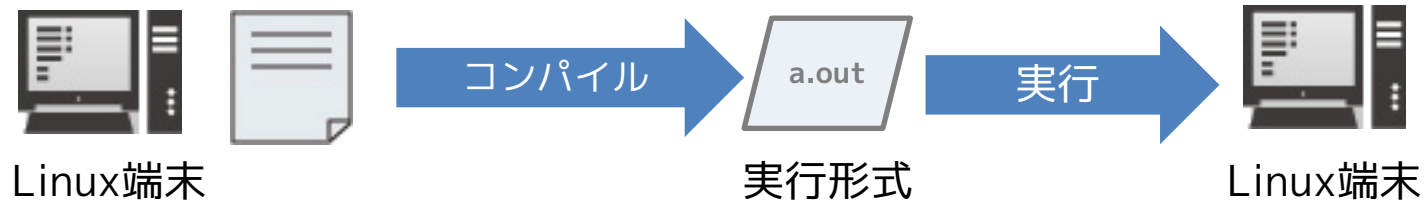
プログラムを「機械が実行できる形式」に変換すること



# コンパイルの種類

## セルフコンパイル

コンパイルした環境と同じ環境で実行



## クロスコンパイル

コンパイルした環境とは別の環境で実行



当センターでは「クロスコンパイル」を使用



# コンパイルの方法

## コンパイルを行う際のコマンド

|                          | Fortran言語 | C言語   | C++言語 |
|--------------------------|-----------|-------|-------|
| SXクロスコンパイラ<br>(SX-ACE用)  | sxf90     | SXC++ |       |
| Intelコンパイラ<br>(HCC,VCC用) | ifort     | icc   | icpc  |

## コマンド例(SX-ACE用Fortranプログラム)

\$ **sxf90** program.f

→実行形式ファイル「a.out」が生成

# コンパイルオプション

コンパイル時にオプションを指定することで  
様々な機能を使用することが可能

```
$ sxf90 program.f -option
```

オプションの一例

**-o [filename]** : 実行形式のファイル名を指定  
指定しない場合は「a.out」が出力

**-Rn** : 翻訳リスト出力（ nには0～5を指定）  
最適化等によるプログラムの変形内容を出力

**-ftrace** : 簡易性能解析機能  
ジョブスクリプトに“setenv F\_FTRACE YES ”の指定が必要  
プログラム実行後に解析ファイルを出力

# コンパイルオプション(参考)

## オプションの一例

### -P [suboption] : 並列化オプション

並列化処理を使用する場合に指定

suboptionには、auto、openmp、multi等を指定可能

### -C [suboption] : 最適化オプション

ベクトル化、最適化のレベル指定

suboptionには、hopt、vopt、vsafe、ssafe、debugを指定可

詳しい解説は下記の講習会にて行います

**SX-ACE 高速化技法の基礎**

日時： 2016年9月8日(木) 13:30 - 17:30

# 演習1（コンパイル）

1. 演習用プログラムを取得してください

(例) \$ cp /sc/cmc/apl/kousyu/nyumon/sample.f ~/

2. sample.f をSX用にコンパイルしてください

(例) \$ sxf90 -o sx.out sample.f

3. sample.f をHCC,VCC用にコンパイルしてください

(例) \$ ifort -o pcc.out sample.f

※文字入力時は [Tab]キーでの補完機能を活用してください

# 本日のプログラム

I. システムのご紹介

## II. 利用方法の解説・実習

i. システムへの接続

ii. プログラムの作成・コンパイル

iii. ジョブスクリプトの作成

iv. ジョブスクリプトの投入

III. 利用を希望する方へ

# 計算機の利用方法

## 会話型（インタラクティブ利用）

コマンド等を通してコンピュータに直接命令し、リアルタイムで処理を実行

操作として手軽

## 一括処理型（バッチ利用）

コンピュータにまとめて処理を命令し実行

処理の命令が終われば、ログアウトしてもOK

# 会話型

原則として利用不可

旧SXでは会話型が利用可能だった

→**SX-ACEでは利用不可**

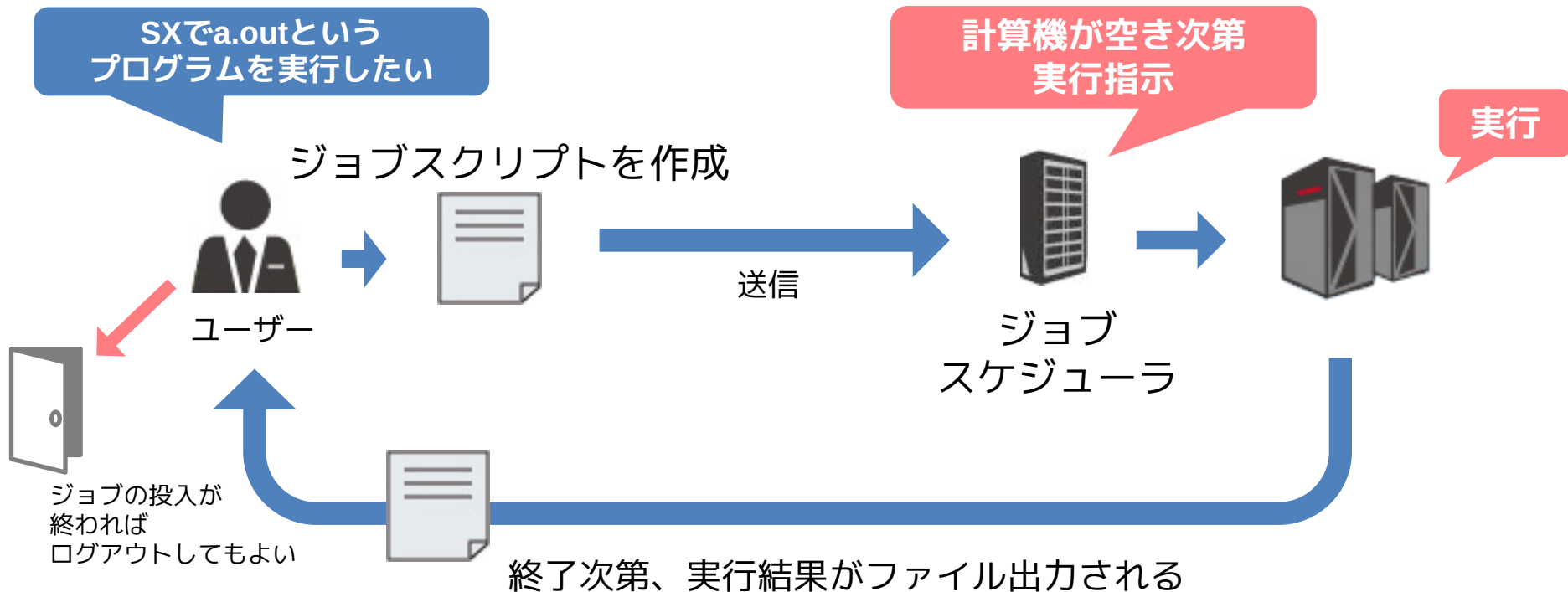
ただし会話型風に一括処理利用する機能はあり

フロントエンド端末での計算実行も禁止

基本的に「一括処理型」で利用

# 一括処理型

処理を「ジョブスクリプト」に記述  
スクリプトに基づき計算機が処理を実行





# ジョブスクリプト

## ジョブスクリプトの構成

リソースや環境設定：#PBSから始まるNQSオプション  
計算機に実行させる処理の記述：シェルスクリプト

## ジョブスクリプトの例

```
#!/bin/csh
```

リソース、環境設定の指定

```
#PBS -q ACE
```

```
#PBS -l elapstim_req=1:00:00,memsz_job=60GB
```

```
cd $PBS_O_WORKDIR
```

```
./a.out > result.txt
```

計算機に実行させる処理の記述

# リソース、環境設定の指定

NQSIIオプション(以下)でリソースや環境の設定を行う

| オプション   | 説明                               |
|---------|----------------------------------|
| #PBS -q | ジョブクラスを指定し、計算に使用する計算機やリソースを指定する  |
| #PBS -l | 使用する資源値                          |
|         | memsz_job : 1ノードあたりのメモリ量         |
|         | elapstim_req : ジョブの経過時間          |
|         | cpunum_job : 1ノード当たりのCPU数        |
| #PBS -m | 計算の処理状態に変化が起きたときメール通知を行う         |
|         | a : ジョブが異常終了したとき                 |
|         | b : ジョブが開始したとき                   |
|         | e : ジョブが終了したとき                   |
| #PBS -M | メールの通知先アドレスを指定する                 |
| #PBS -v | 環境変数の指定(setenvではなくこちらを使うことを推奨する) |
| #PBS -T | MPI 実行時に指定                       |
|         | mpisx : MPI/SX 利用時               |
|         | intmpi : IntelMPI 利用時            |
| #PBS -b | 使用するノード数                         |

必須！

# ジョブクラス一覧(SX-ACE)

使用する計算機、リソースはジョブクラスで指定  
NQSIIオプション「#PBS -q」の後に続けて記述

| ジョブクラス | 利用可能<br>経過時間 | 利用可能<br>最大Core数            | 利用可能<br>メモリ            | 同時利用可能<br>ノード数 |
|--------|--------------|----------------------------|------------------------|----------------|
| ACE    | 24時間         | 1024Core<br>(4Corex256ノード) | 1.5TB<br>(60GBx256ノード) | 256ノード         |
| DBG    | 20分          | 32Core<br>(4Corex8ノード)     | 480GB<br>(60GBx8ノード)   | 8ノード           |

# ジョブクラス一覧(HCC)

| ジョブクラス   | 利用可能<br>経過時間  | 利用可能<br>CPU数           | 利用可能<br>メモリ           | 同時利用<br>可能ノード数 |
|----------|---------------|------------------------|-----------------------|----------------|
| H-single | 最大300時間<br>程度 | 2CPU<br>(2CPU×1ノード)    | 4GB<br>(4GB×1ノード)     | 1ノード           |
| H-small  | 最大300時間<br>程度 | 32CPU<br>(2CPU×16ノード)  | 64GB<br>(4GB×16ノード)   | 2～16ノード        |
| H-large  | 最大300時間<br>程度 | 128CPU<br>(2CPU×64ノード) | 256GB<br>(4GB×64ノード)  | 17～64ノード       |
| H-mem+   | 最大300時間<br>程度 | 64CPU<br>(2CPU×32ノード)  | 384GB<br>(12GB×32ノード) | 1～32ノード        |

# ジョブクラス一覧(VCC)

| ジョブクラス         | 利用可能<br>経過時間 | 利用可能最大<br>Core数           | 利用可能<br>メモリ            | 同時利用<br>可能ノード数 |
|----------------|--------------|---------------------------|------------------------|----------------|
| VCC            | 120時間        | 640Core<br>(20Corex32ノード) | 1920GB<br>(60GBx32ノード) | 32ノード          |
|                | 336時間        | 40Core<br>(20Corex2ノード)   | 120GB<br>(60GBx2ノード)   | 2ノード           |
| GVC<br>(GPU利用) | 120時間        | 180Core<br>(20Corex9ノード)  | 540GB<br>(60GBx9ノード)   | 9ノード           |

# 計算機に実行させる処理の記述

ファイルやディレクトリの実行・操作を記述  
記述方法はシェルスクリプト

よく使用するNQSII 用の環境変数

**\$PBS\_O\_WORKDIR** : ジョブ投入時のディレクトリが設定される

標準出力／標準エラー出力の容量制限

⇒ SX-ACE : 100MB、VCC:100MB、HCC:10MB

⇒ これ以上出力したい場合はリダイレクション(>)を利用

処理の記述の最終行に改行を入れること！

⇒ 未入力の場合、その行のコマンドが実行されない

# ジョブスクリプト解説

ジョブクラスの指定

```
#!/bin/csh
```

```
#PBS -q ACE
```

CPU数、経過時間、メモリサイズの指定  
コマ後にスペースを入れないよう注意！

```
#PBS -l elapstim_req=1:00:00,memsz_job=60GB
```

```
cd $PBS_O_WORKDIR
```

ジョブ投入時のディレクトリへ移動

```
./a.out > result.txt
```

a.outを実行し、結果をresult.txtに出力する  
(リダイレクション)

# 演習2（ジョブスクリプト）

1. 演習用スクリプトを取得してください

(例)\$ [cp /sc/cmc/apl/kousyu/nyumon/sample.nqs ~/](#)

2. sample.nqsを元にSX-ACE用のジョブスクリプトを作成してください

(例) \$ [cp sample.nqs sx.nqs](#)

\$ [emacs sx.nqs -nw](#)

ジョブクラスはDBGを使用してください



# 本日のプログラム

I. システムのご紹介

## II. 利用方法の解説・実習

i. システムへの接続

ii. プログラムの作成・コンパイル

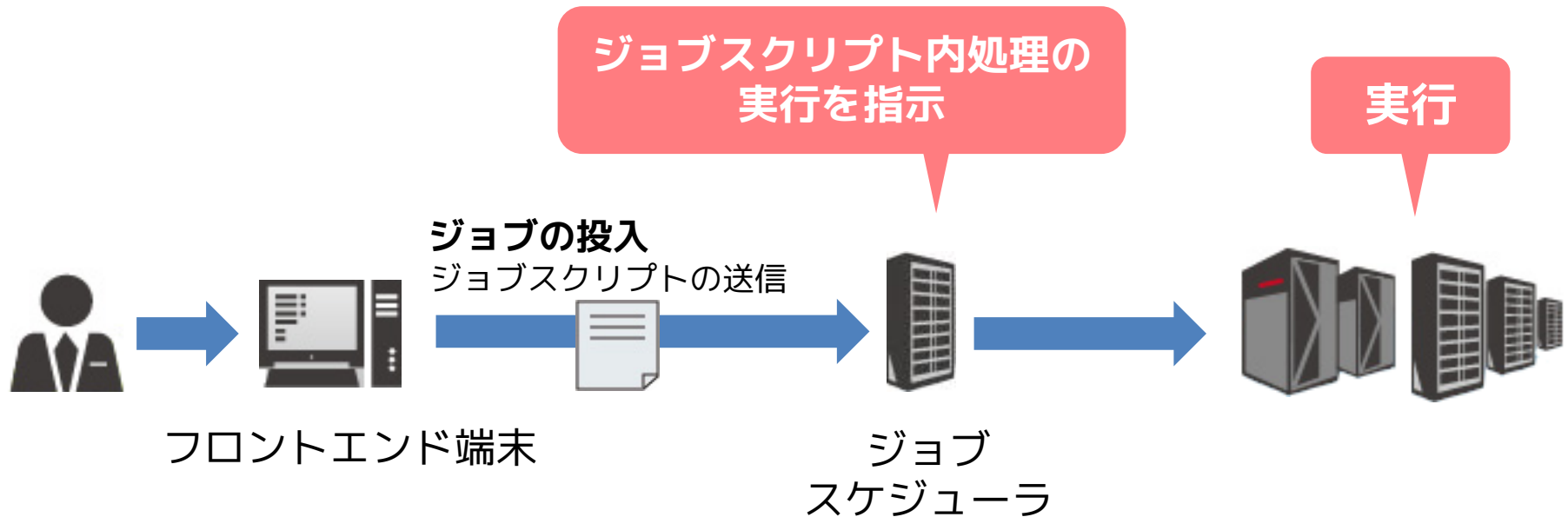
iii. ジョブスクリプトの作成

iv. ジョブスクリプトの投入

III. 利用を希望する方へ

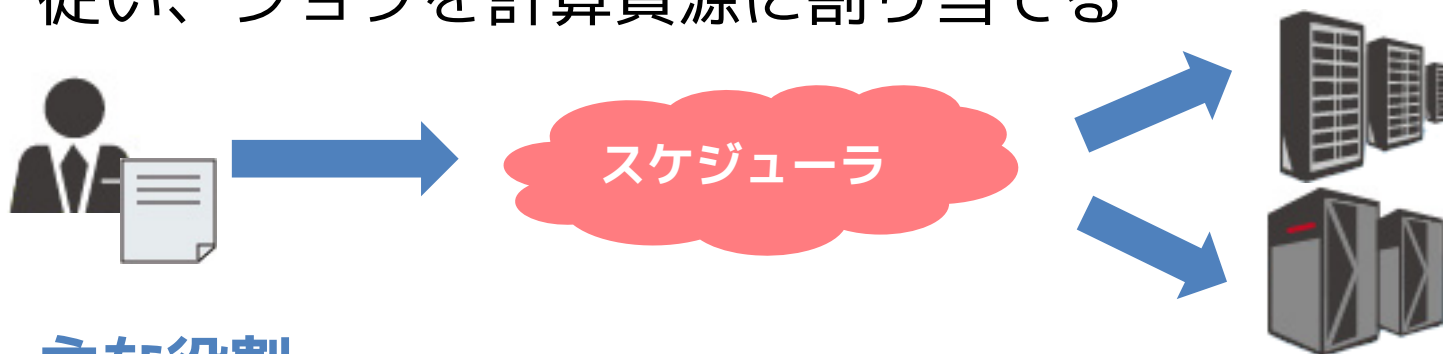
# 実行までの流れ

ジョブスクリプトは**ジョブスケジューラ**が受け付ける  
ジョブスケジューラが各計算機にジョブの実行を指示



# スケジューラとは

あらかじめ管理者によって設定された資源割当ポリシーに従い、ジョブを計算資源に割り当てる



## 主な役割

クラスタを構成する計算機(ノード)の静的情報※を把握

※ディスク容量、メモリ容量、CPU性能、etc

ノード毎の資源使用率を定期的に監視、管理

ユーザより実行したいジョブ要求を受信

ジョブを実行するのに適切なノードを選定

ジョブ実行に伴う入出力データのファイル転送

# スケジューラとは

当センターではバックフィル型を採用

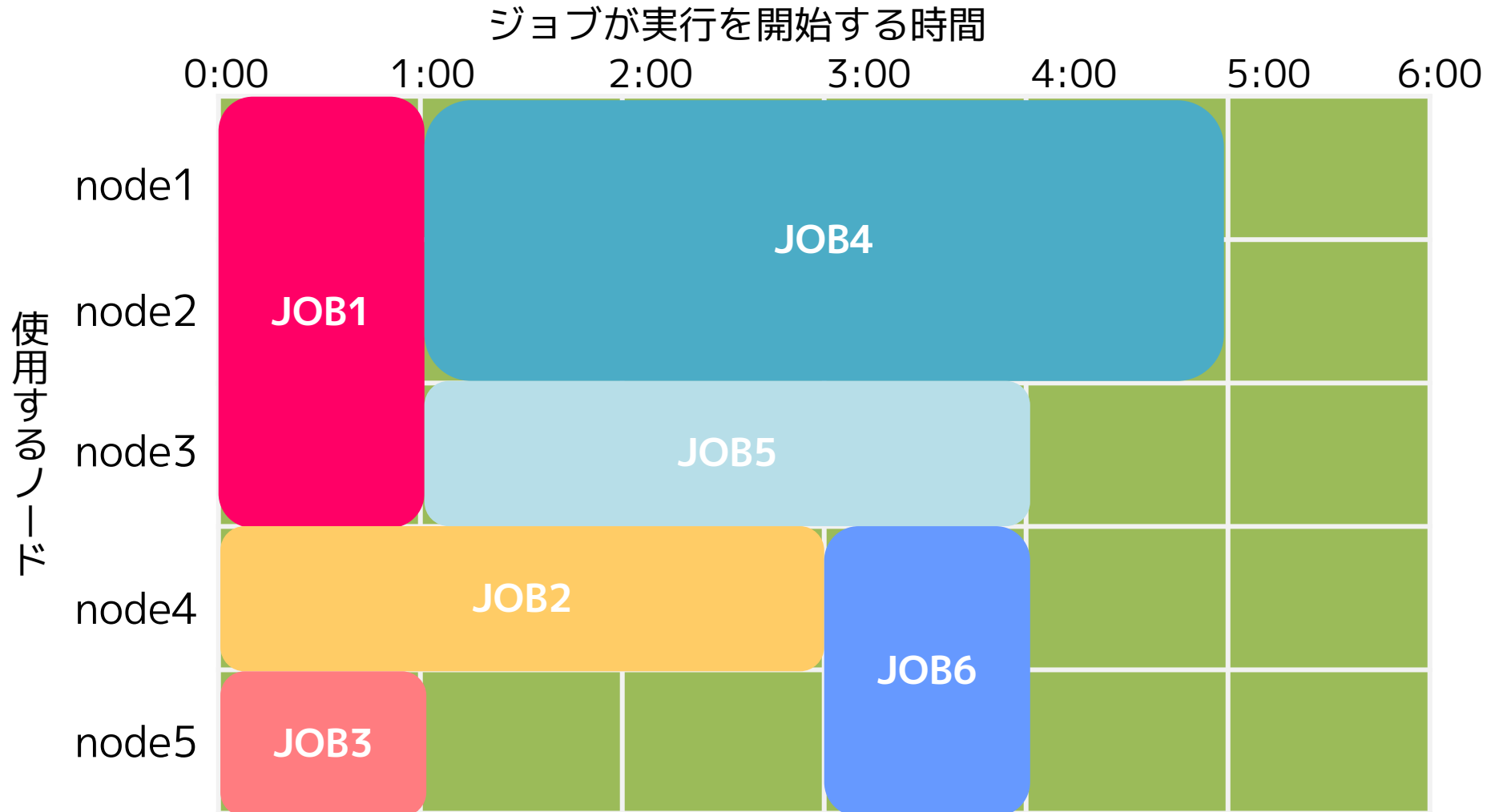
## 特徴

ジョブの実行開始時間のマップを作成する

マップに載れば、実行開始時間と経過時間が保障される

実行中は指定したリソースを占有して割当ててる

# スケジューラのイメージ



# ジョブの投入方法

フロントエンド端末からジョブを投入  
コマンド

```
$ qsub [ジョブスクリプトファイル]
```

(参考) 特殊な投入方法

リクエスト連携：順番通りにジョブを実行したい場合に利用

```
$ qsub [JobScript1][JobScript2] . . .
```

※ 順番無く複数のジョブを同時に投入する場合は  
上記のようにしないよう注意

# 投入済みジョブの確認方法

ジョブの状態を確認することが可能

コマンド

\$ **qstat**

実行結果

| RequestID | ReqName  | UserName | Queue | STT | Memory | CPU    | Elapse |
|-----------|----------|----------|-------|-----|--------|--------|--------|
| 12345.cmc | nqs-test | a61234   | ACE   | RUN | 8.72G  | 830.66 | 208    |

## ジョブの状態

待ち状態では「QUE」 実行が始まると「RUN」となる。

## 実行時間

CPU : 実際にジョブが消費した時間  
複数CPU指定の場合は、全CPUを累積表示

Elapse : ジョブが実行されてからの経過時間

# 投入済みジョブの確認方法

ジョブの予約状況の確認することが可能

コマンド

\$ **sstat**

実行結果

| RequestID | ReqName  | UserName | Queue | Pri              | STT | PlannedStartTime    |
|-----------|----------|----------|-------|------------------|-----|---------------------|
| 12345.cmc | nqs-test | a61234   | ACE   | -1.5684/ -1.5684 | ASG | 2015-06-16 00:01:23 |

## 状態監視

実行時刻が決まると「ASG」表示になる。

混雑具合や優先度により、「実行時間の決定」までの待ち時間が異なるが、一旦実行時間が決定されるとその時刻にジョブ実行が始まる。

## 実行開始時刻

システムメンテナンスやトラブル時は再スケジュールされることをご了承ください。



# 投入済みジョブの操作方法

ジョブのキャンセル

コマンド

```
$ qdel [RequestID]
```

実行結果

```
$ qdel 12345.cmc
```

```
Request 12345.cmc was deleted.
```

# 実行結果の確認方法

実行結果や実行エラーは指定しない限り「標準出力」となる

標準出力はジョブスクリプト名.oリクエストID  
標準エラー出力はジョブスクリプト名.eリクエストID  
というファイル名で自動出力される

catやlessコマンドでファイルの内容を出力し確認

```
$ cat nqs.o12345
```

※リダイレクション（./a.out > result.txt）を使った場合は、そちらも確認

意図通りの結果が表示されていれば計算は成功

# 演習3（ジョブスクリプトの投入）

1. 作成したジョブスクリプトを使用してジョブを投入

`$ qsub sx.nqs`

2. 投入したジョブの状態を確認

`$ sstat`

`$ qstat`

3. 結果ファイルの確認

`$ cat sx.nqs.o12345`

`$ cat sx.nqs.e12345`

早く終わった方はVCCやHCCにも  
ジョブを投入してみましょう

# より高度な利用に向けて

## 利用の参考になるWebページ

サイバーメディアセンター 大規模計算機システム Webページ  
<http://www.hpc.cmc.osaka-u.ac.jp/system/manual/>

### 利用方法

<http://www.hpc.cmc.osaka-u.ac.jp/system/manual/>

### FAQ

<http://www.hpc.cmc.osaka-u.ac.jp/faq/>

### お問い合わせ

[http://www.hpc.cmc.osaka-u.ac.jp/support/contact/auto\\_form/](http://www.hpc.cmc.osaka-u.ac.jp/support/contact/auto_form/)

### 研究成果

<http://www.hpc.cmc.osaka-u.ac.jp/researchlist/>

# 本日以降に実施予定の講習会



**SX-ACE  
向け**

**9月8日(木) 13:30-17:30**  
SX-ACE高速化技法の基礎



**VCC, HCC  
向け**

**9月9日(金) 13:30-16:30**  
並列コンピュータ高速化技法の基礎

**9月15日(木) 10:30-16:30**  
SX-ACE並列プログラミング入門(MPI)

**9月16日(金) 13:30-16:30**  
SX-ACE並列プログラミング入門(HPF)

**毎年2回（6月/9月）開催！**

# 本日のプログラム

I. システムのご紹介

II. 利用方法の解説・実習

i. システムへの接続

ii. プログラムの作成・コンパイル

iii. ジョブスクリプトの作成

iv. ジョブスクリプトの投入

**III. 利用を希望する方へ**

# 利用を希望する方へ

本センターの大規模計算機システムは  
どなたでも**利用可能**です！

大学院生

教員

研究者

大阪大学

他大学

民間企業

利用負担金が必要になります

# 利用負担金



**SX-ACE**

共有利用



**SX-ACE**

占有利用



**VCC**

共有利用



**VCC**

占有利用



ディスク  
容量追加  
オプション  
(1TB単位)



**HCC**

共有利用



HCC&ディスク500GBは  
SX-ACE、VCCの利用者に  
無償で提供しています



# 計算機の提供方法

## 共有利用

「ノード時間」単位で  
ノードを利用

利用者全員で一定数の  
ノードを共有

大規模なノード間並列を  
試せる

「待ち時間」が発生する

## 占有利用

「年度/月」単位で  
ノードを利用

他の利用者のグループと  
ノードを共有しない

大規模なノード間並列は試  
し難い

「待ち時間」が発生しない

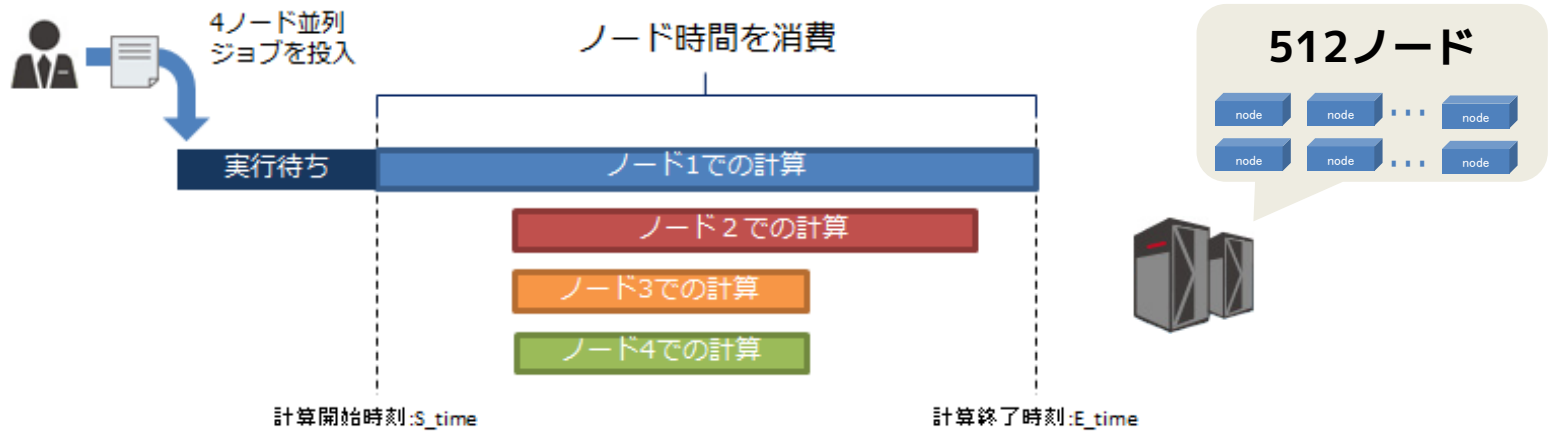
# 「ノード時間」とは

$$\text{ノード時間} = \text{計算に使用するノード数} \times \text{計算時間(単位:時間)}$$

(例)

|               |   |            |
|---------------|---|------------|
| 1ノードで3時間の計算   | → | 3ノード時間消費   |
| 30ノードで5時間の計算  | → | 150ノード時間消費 |
| 100ノードで1時間の計算 | → | 100ノード時間消費 |
| 1ノードで100時間の計算 | → | 100ノード時間消費 |

# 「ノード時間」とは



ノード時間は4ノード × (計算終了時間 - 計算開始時間)です



ノード内で使用するコアを限定しても、ノード時間は変わりません

# 利用可能なアプリケーション

AVS/Express \*  
IDL \*

Gaussian09 \*  
GROMACS  
LAMMPS  
OpenFOAM  
(VCCで提供)

MSC Software

- Marc / Mentat \*
- Dytran \*
- Patran \*
- Adams \*
- Nastran \*

(フロントエンド端末で提供)

\*のついたアプリケーションは  
学内向けにメディア貸出サービスを実施しています (計算機利用者以外も可)

# まずは試用制度をお試ください

3 カ月間 以下の資源をご提供

**無料**



**SX-ACE**

共有利用

500ノード時間



**VCC**

共有利用

500ノード時間



ディスク500GB付き！

全てのアプリケーションも利用可能です

# 利用申請方法

大規模計算機システムの利用申請は  
**随時受け付け中**です！

申請は年度単位(4月から翌年3月まで)です  
申請はWEBフォームから受け付けています

詳細は下記のページをご覧ください！

一般利用(学術利用) <http://osku.jp/u094>

試用制度による利用 <http://osku.jp/e029>

**大規模計算機システムに関するご質問は**

**大阪大学 情報推進部 情報基盤課  
研究系システム班**

**system@cmc.osaka-u.ac.jp**

**までお気軽にご連絡ください！**