

# スパコンの利用方法

大阪大学 情報推進部 情報基盤課

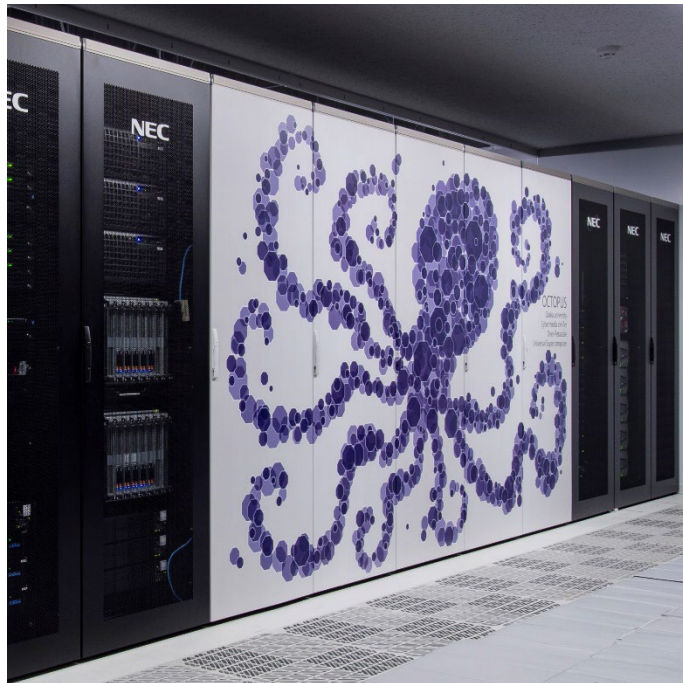
# 本日のプログラム

- I. システムのご紹介
- II. 利用方法の解説
  - i. システムへの接続
  - ii. プログラムの作成・環境設定・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

# OCTOPUS

2017年12月運用開始

構成の異なる4種類のノードで構成されている



|      | CPUノード          | GPUノード           | Xeon Phiノード     | 大容量主記憶<br>搭載ノード |
|------|-----------------|------------------|-----------------|-----------------|
| コア数  | 24              | 24               | 64              | 128             |
| 演算性能 | 1.996<br>TFLOPS | 23.196<br>TFLOPS | 2.662<br>TFLOPS | 8.192<br>TFLOPS |
| メモリ  | 192GB           | 192GB            | 192GB           | 6TB             |
| ノード数 | 236ノード          | 37ノード            | 44ノード           | 2ノード            |

合計319ノード 1.463PFLOPS

# SQUID

2021年5月運用開始

構成の異なる3種類のノードで構成されている



|      | CPUノード          | GPUノード                | ベクトルノード              |
|------|-----------------|-----------------------|----------------------|
| コア数  | 76              | 76                    | VH:64<br>VE:10       |
| 演算性能 | 5.837<br>PFLOPS | 161.837<br>PFLOPS     | 24.56<br>PFLOPS      |
| メモリ  | 256GB           | 512GB                 | VH:128GB<br>VE:48GB  |
| ノード数 | 1520ノード         | 42ノード<br>(8GPU / ノード) | 36ノード<br>(8VE / ノード) |

合計1,598ノード 16.591PFLOPS

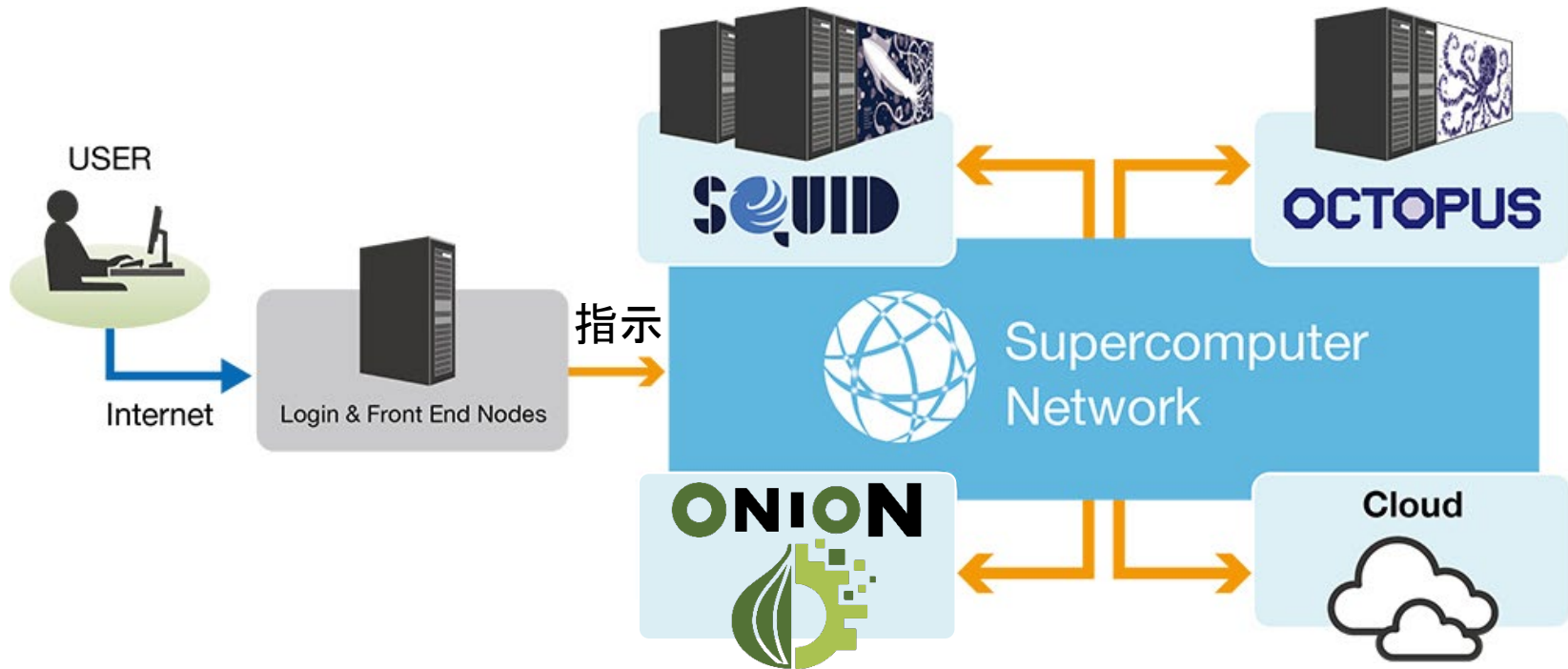
# フロントエンドサーバ

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  
作業用サーバ

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

スパコン本体へのログインは原則禁止(一部例外有)

# システム全体図



今回はSQUID、OCTOPUSの使い方を紹介します

# 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

- i. システムへの接続
- ii. プログラムの作成・環境設定・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

# 利用の流れ

ユーザー

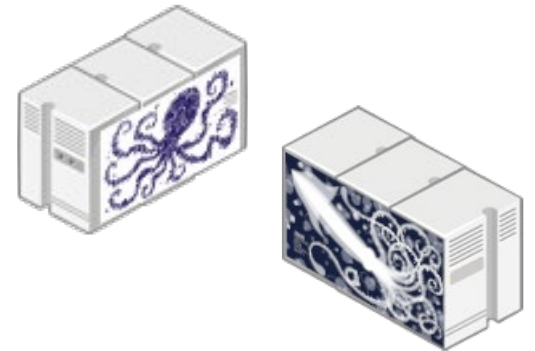


フロントエンド端末



システムへの接続

計算機



環境設定

プログラム作成

コンパイル

ジョブスクリプト  
作成

ジョブスクリプト  
投入



# 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

i. システムへの接続

ii. プログラムの作成・環境設定・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

# システムへの接続

ログインはSSH (Secure Shell) 接続

Win: コマンドプロンプト、TeraTermなど Mac: ターミナル

接続先は OCTOPUS : [octopus.hpc.cmc.osaka-u.ac.jp](https://octopus.hpc.cmc.osaka-u.ac.jp)  
SQUID : [squidhpc.hpc.cmc.osaka-u.ac.jp](https://squidhpc.hpc.cmc.osaka-u.ac.jp)

ユーザー



フロントエンド端末



複数台あり、ログインユーザの  
少ない端末に自動で振り分ける

# SQUIDへのログイン

SQUIDは二段階認証でのログインとなります  
OCTOPUSと異なり二段階認証用の端末が必要です

アカウントとパスワード



二段階認証用の端末(スマホ or PC)



# 二段階認証用の端末

ご自身のスマートフォンやパソコンを  
二段階認証用の端末としてお使いください

以下いずれかのアプリケーションをインストールしてください

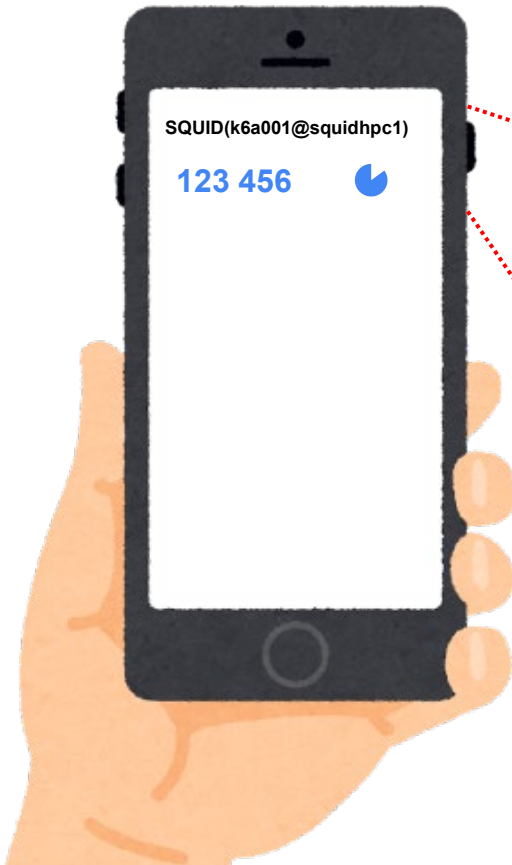
| OS      | アプリケーション                | 配布元                               |
|---------|-------------------------|-----------------------------------|
| Android | Google Authenticator    | <a href="#">Google Play Store</a> |
| iOS     | Microsoft Authenticator | <a href="#">Apple App Store</a>   |
| Windows | WinAuth                 | <a href="#">Github</a>            |
| macOS   | Step Two                | <a href="#">Apple App Store</a>   |



# SQUIDへのログイン

初回のみ

SQUIDに初めてログインするとQRコードが表示されます。  
QRコードをアプリで読み込むことで2段階認証の登録が完了します



```
Initialize google-authenticator

Warning: pasting the following URL into your browser exposes
the OTP secret to Google:

https://www.google.com/chart?chs=200*200&chld=M|0&cht=q
r&otpauth://totp/user1@squidhpc.hpc.cmc.osaka-
u.ac.jp%3Fsecret%3DXXXXXXXXXXCLI%26issuer%3Dsquidh
pc.hpc.cmc.osaka-u.ac.jp

QR
コード

Your new secret key is: XXXXXXXXXXXXX

Enter code from app (-1 to skip): -1
Code confirmation skipped
Your emergency scratch codes are:
```

# 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

i. システムへの接続

ii. プログラムの作成・環境設定・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

当センターの計算機で使用可能な主な言語

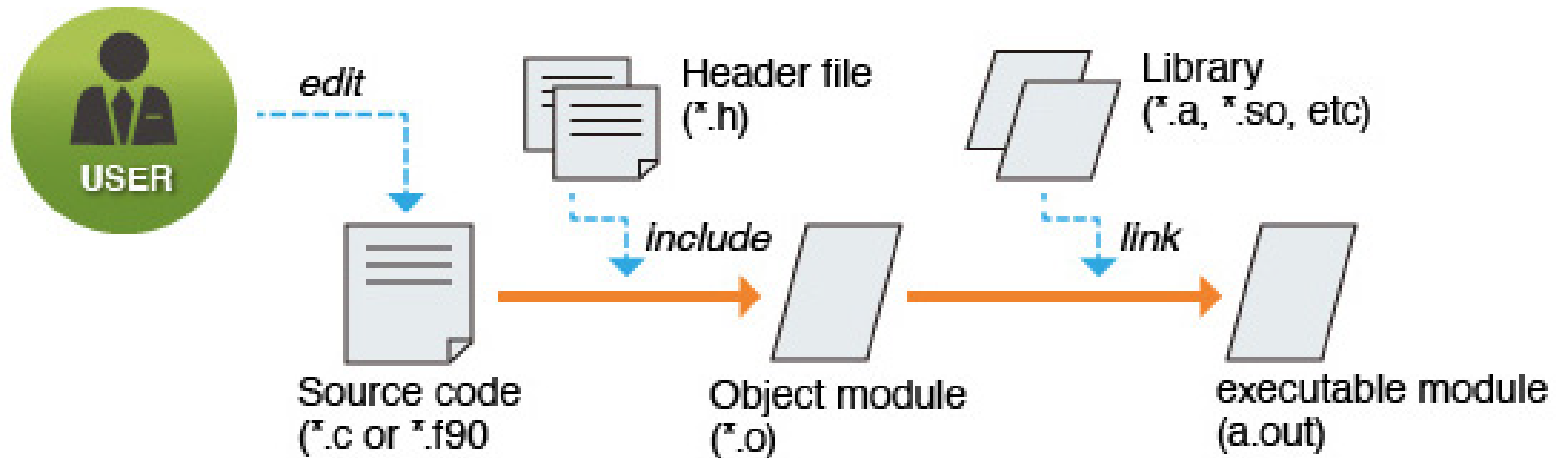
Fortran言語、C言語、C++言語

Python、R、Julia、etc...

「プログラムの書き方」については  
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





# 環境設定 (OCTOPUSでは不要)

利用するノード群に応じて環境設定が必要  
モジュールを読み込むことで一括設定が可能

|       | 汎用CPUノード     | GPUノード       | ベクトルノード      |
|-------|--------------|--------------|--------------|
| モジュール | BaseCPU/2023 | BaseGPU/2023 | BaseVEC/2023 |

コマンド例: 汎用CPUノードを利用する場合

```
$ module load BaseCPU/2023
```

# コンパイルの方法

## コンパイルを行う際のコマンド(SQUID)

|                              | Fortran言語 | C言語 | C++言語 |
|------------------------------|-----------|-----|-------|
| Intelコンパイラ<br>(汎用CPUノード向け)   | ifort     | icc | icpc  |
| NVIDIA HPC SDK<br>(GPUノード向け) | nvfortran | nvc | nvc++ |
| NECコンパイラ<br>(ベクトルノード向け)      | nfort     | ncc | nc++  |

コマンド例:汎用CPUノード用のFortranプログラム

```
$ ifort 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

```
$ ifort 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

**-parallel** : 並列化オプション

並列化処理を使用する場合に指定

**-O...** : 最適化オプション

最適化のレベル指定

# 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

i. システムへの接続

ii. プログラムの作成・環境設定・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

# 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

操作として手軽で直感的

## 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

命令が終われば、放置（ログアウト）してもOK

# 会話型

原則として利用不可

現在稼働中の計算機では利用不可

ただし“会話型風”の機能(インタラクティブ利用)はあり

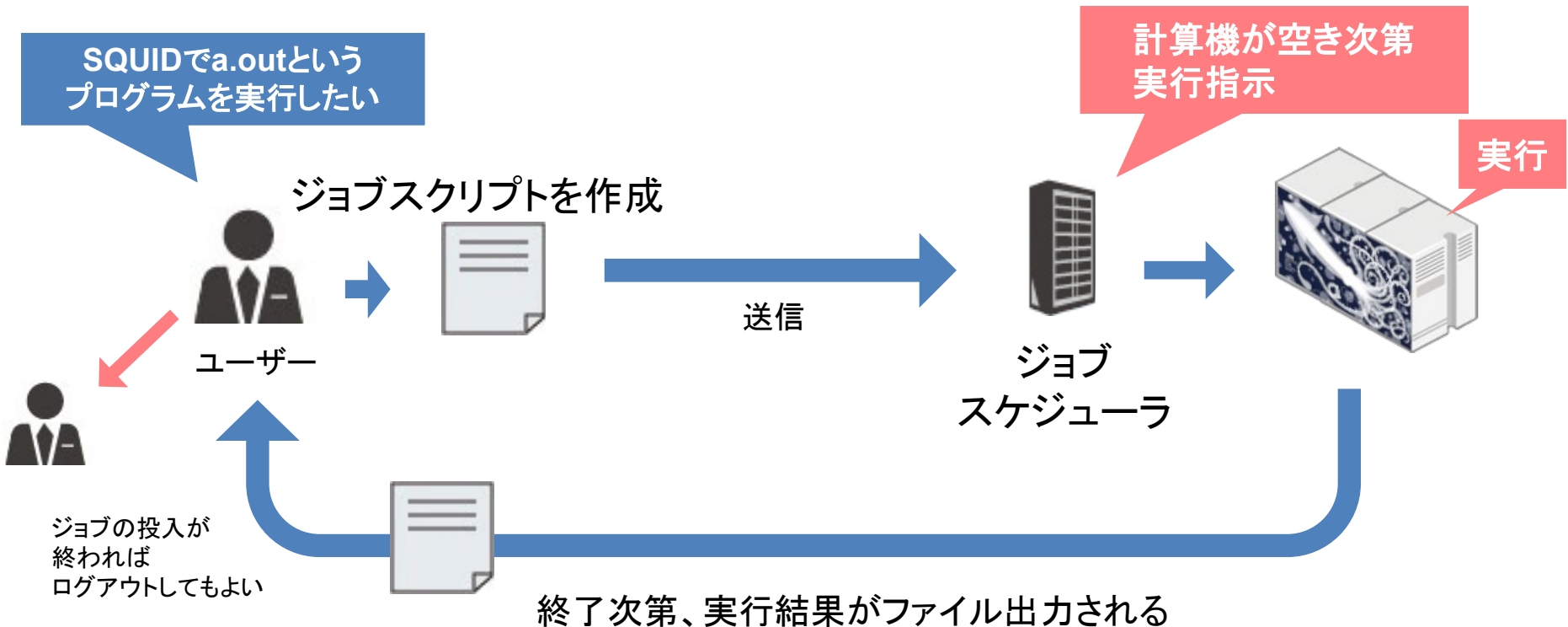
フロントエンド端末での計算実行も禁止

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

ジョブ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/bash
```

リソース、環境設定の指定

```
#PBS -q SQUID
```

```
#PBS -l elapstim_req=1:00:00
```

```
#PBS --group=G12345
```

```
module load BaseCPU/2023
```

```
cd $PBS_O_WORKDIR
```

```
./a.out
```

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

NQS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                          |
|               | elapstim_req : ジョブの経過時間          |
|               | memsz_job : 1ノードあたりのメモリ量         |
|               | cpunum_job : 1ノード当たりのコア数         |
|               | gpunum_job : 1ノード当たりのGPU数        |
| #PBS --venode | 1ノードあたりのVE数                      |
| #PBS --group  | グループの指定                          |
| #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(exportではなくこちらを使うことを推奨する) |

必須!

場合により  
必須

# ジョブクラス一覧(OCTOPUS)

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

| ジョブクラス  | 対象ノード              | 利用可能経過時間 | 利用可能コア数                      | 利用可能メモリ                    | 同時利用可能ノード数 |
|---------|--------------------|----------|------------------------------|----------------------------|------------|
| OCTOPUS | 汎用CPUノード<br>GPUノード | 120時間    | 3,072Core<br>(24Core×128ノード) | 23,680GB<br>(185GB×128ノード) | 128ノード     |
| OCTPHI  | Xeon Phiノード        | 120時間    | 2,048Core<br>(64Core×32ノード)  | 6,080GB<br>(185GB×32ノード)   | 32ノード      |
| DBG     | 汎用CPUノード<br>GPUノード | 10分      | 24Core<br>(24Core×1ノード)      | 185GB<br>(185GB×1ノード)      | 1ノード       |

# ジョブクラス一覧(SQUID)

使用する計算機、リソースはジョブクラスで指定  
NQS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>コア数                   | 利用可能<br>メモリ             | 同時利用<br>可能ノード数 | 備考    |
|---------|--------------|-------------------------------|-------------------------|----------------|-------|
| SQUID   | 120時間        | 38,912Core<br>(76Core×512ノード) | 124TB<br>(248GB×512ノード) | 512ノード         |       |
| SQUID-H | 120時間        | 38,912Core<br>(76Core×512ノード) | 124TB<br>(248GB×512ノード) | 512ノード         | 高優先度  |
| SQUID-S | 120時間        | 38Core<br>(76Core×0.5ノード)     | 124GB<br>(248GB×0.5ノード) | 0.5ノード         | ノード共有 |
| DBG     | 10分          | 152Core<br>(76Core×2ノード)      | 496GB<br>(248GB×2ノード)   | 2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NQS 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

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/bash
```

```
#PBS -q SQUID
```

ノードを1時間確保

```
#PBS -l elapstim_req=1:00:00
```

```
#PBS --group=G12345
```

```
module load BaseCPU/2023
```

汎用CPUノードの環境設定を読み込み

```
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)

# 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

i. システムへの接続

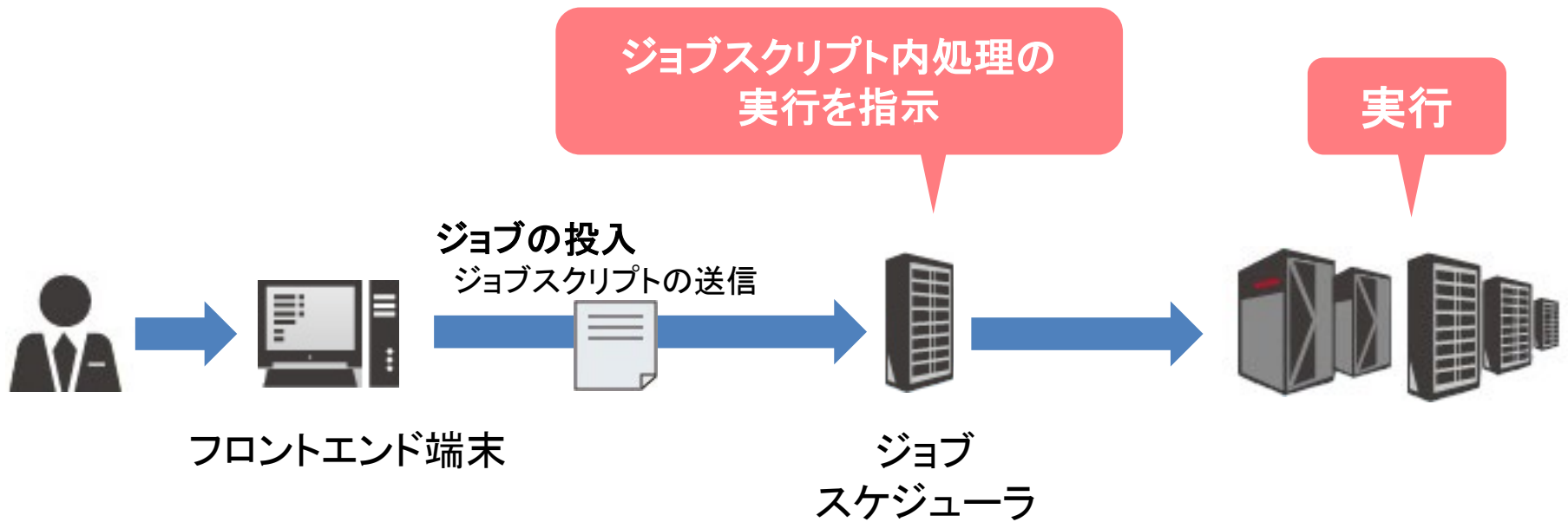
ii. プログラムの作成・環境設定・コンパイル

iii. ジョブスクリプトの作成

iv. ジョブスクリプトの投入

# 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

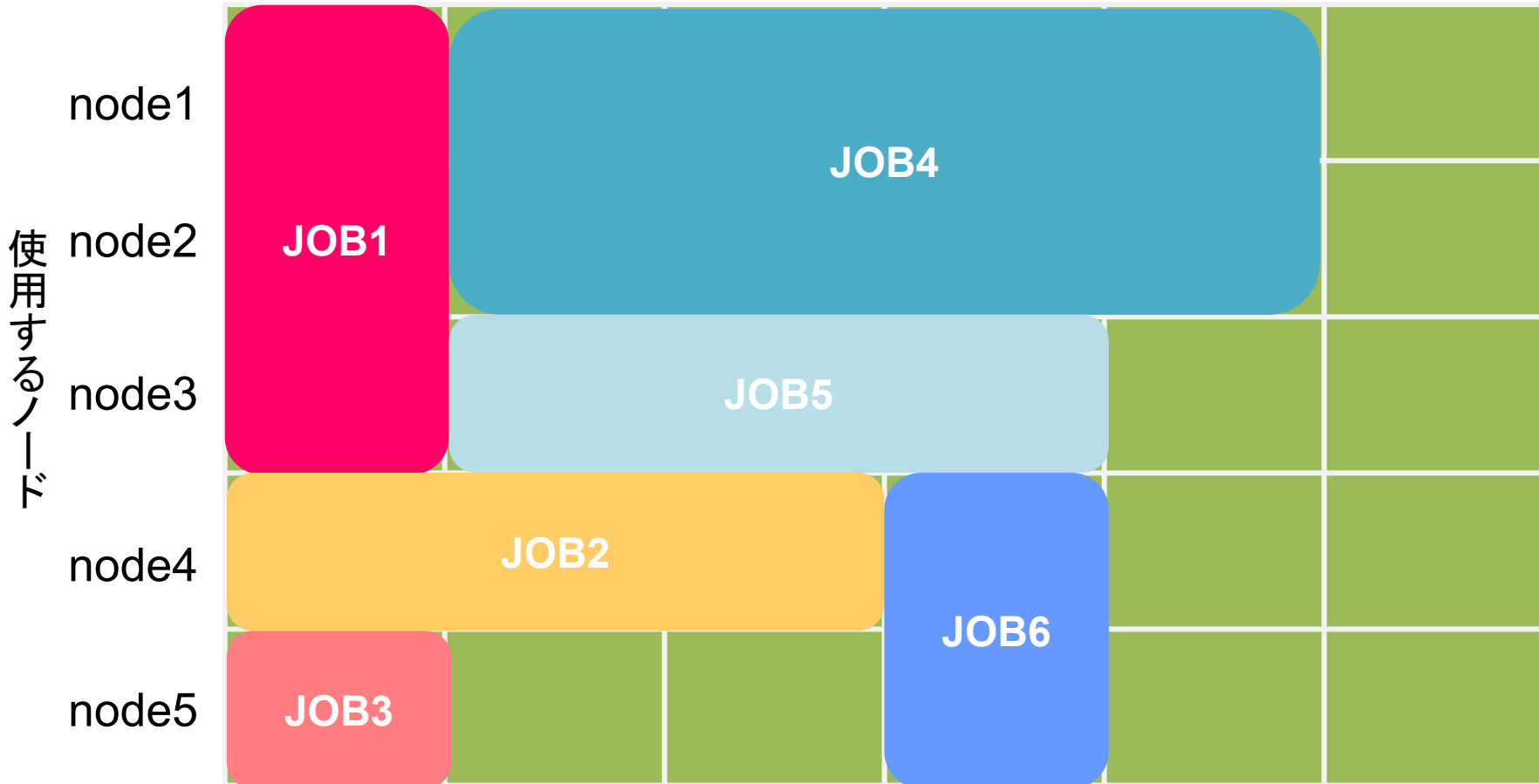
マップに載れば、実行開始時間と経過時間が保障される

実行中は指定したリソースを占有して割り当てる

# スケジューラのイメージ

ジョブが実行を開始する時間

0:00 1:00 2:00 3:00 4:00 5:00 6:00



# 「ノード時間」とは

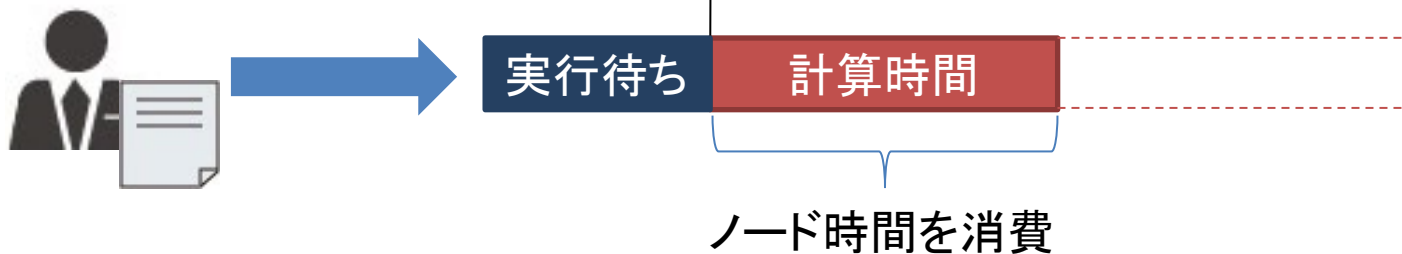
$$\text{ノード時間} = \text{計算に使用するノード数} \times \text{計算時間(単位:時間)}$$

(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」の注意点

```
#!/bin/bash  
#PBS -q SQUID  
#PBS -l elapstim_req=1:00:00
```



消費するノード時間は、実際にかかった計算時間のみです

# 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

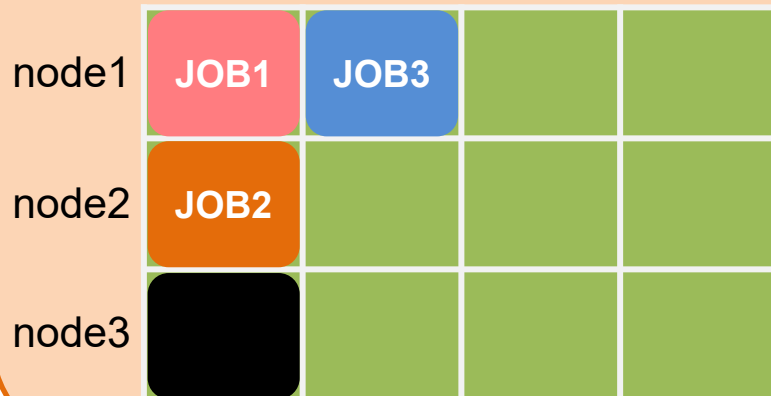
コマンド

```
$ qsub [ジョブスクリプトファイル]
```

(参考)複数ジョブを投入する場合

順不同でスケジューリング

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



順番通りにスケジューリング

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



# 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 |
|-----------|----------|----------|-------|-----|--------|--------|--------|
| 1234.sqd  | job-test | K6a001   | SC1   | 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

ジョブの予約状況を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    |
|-----------|----------|----------|-------|------------------|-----|---------------------|
| 1234.sqd  | job-test | k6a001   | SQUID | -1.5684/ -1.5684 | ASG | 2021-05-21 14:52:50 |

## 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 1234.sqd
```

```
Request 1234.sqd 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.sh.o1234
```

※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

標準出力／標準エラー出力の容量制限

⇒ 100MB以上出力したい場合はリダイレクション(>)

# スパコン利用に向けて

## 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>

### 利用方法

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

### FAQ

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

### 問い合わせフォーム

[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/>

# 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

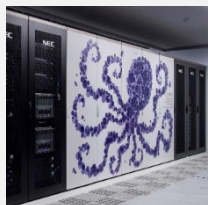
民間企業

利用負担金が必要になります

# 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



**OCTOPUS**

共有利用

26 OCTOPUSポイント  
+ ディスク 3TB

250 ノード時間

60 ノード時間

311 ノード時間

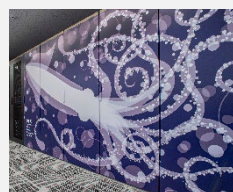
35 ノード時間

汎用  
CPU  
ノード

Xeon  
Phi  
ノード

GPU  
ノード

大容量  
主記憶  
搭載  
ノード



**SQUID**

共有利用

75 SQUIDポイント  
+ ディスク 5TB

250 ノード時間

66 ノード時間

41 ノード時間

汎用  
CPU  
ノード

GPU  
ノード

ベクトル  
ノード

※季節係数、燃料係数が1の場合

# 利用を希望する方へ

## 本日以降の講習会(全てオンライン)

| 講習会名                           | 概要   |
|--------------------------------|--|
| スパコンに通じる並列プログラミングの基礎【5月31日(水)】 | 並列プログラミングの手法や考え方の基礎  |
| 初めてのスパコン【6月5日(月)】              | スーパーコンピュータの基礎的な知識と、その使い方<br>初心者向け・実習あり   |
| OpenMP入門【6月8日(水)】              | OpenMPによる一般的な並列プログラミングの基礎とその<br>利用方法   |
| 並列プログラミング入門<br>(OpenMP/MPI)    | 並列プログラミングの基礎と利用方法  |
| スーパーコンピュータ バッチシステム<br>入門 / 応用  | 計算機利用(バッチ利用)の概要  |
| ONION活用講習会                     | データ集約基盤 ONION(Osaka university Next-generation Infrastructure for Open research and open Innovation) の機能説明 |
| SX-Aurora TSUBASA 高速化技法の<br>基礎 | 性能測定や基礎的なチューニング手法の説明   |

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

または

お問い合わせフォーム

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

までお気軽にご連絡ください！