初めてのスパコン

-ログインから実行まで-

大阪大学 情報推進部 情報基盤課

寺前 勇希

本講習会について

本講習会では**はじめてスパコンを使う方を対象**に
利用方法を解説します

途中、スパコンを利用したデモを行います

配布したアカウントは、講習会後 **1 週間ご利用可能** です
ご自宅からでも接続できますのでご自由にお試しください！

本日のプログラム

1. システムのご紹介
2. 利用方法の解説
   - 2-1 システムへの接続
   - 2-2 プログラムの作成・環境設定・コンパイル
   - 2-3 ジョブスクリプトの作成
   - 2-4 ジョブスクリプトの投入
3. 利用を希望する方へ

システムのご紹介: スパコンって何？

普通のパソコンでは時間のかかる 大規模な計算 を実行するための研究設備

ものすごく速い 1台 のマシンではなく
沢山のサーバを連結している

インターネット経由で
アクセスする

同時に多くの人が使っていて
各々の計算サイズにあわせてサーバを
切り出して使う

システムのご紹介: SQUID

- 3種類の計算ノードと 21 PBのストレージで構成される
- 総理論演算性能は 16.591 PFLOPS
- **2027年6月末でサービス終了予定**

	汎用CPUノード	GPUノード	ベクトルノード
1ノードあたりのコア数	76	76	VH:24 VE:80
1ノードあたりの演算性能	5.837 TFLOPS	161.837 TFLOPS	24.56 TFLOPS
1ノードあたりのメモリ	256GB	512GB	VH:128GB VE:48GB
ノード数 （サーバ数）	1520 ノード	42 ノード （8GPU / ノード）	36 ノード （8VE / ノード）

システムのご紹介: OCTOPUS

- 汎用CPUノード群 と 3.58 PB のストレージで構成される
- 総理論演算性能は 2.293 PFLOPS
- SQUID 汎用CPUノード群と比較し、**1ノードあたりの性能が3倍以上**に

	OCTOPUS 汎用CPUノード	SQUID 汎用CPUノード
1ノードあたりの コア数	256	76
1ノードあたりの 演算性能	16.384 TFLOPS	5.837 TFLOPS
1ノードあたりの メモリ	768 GB	256 GB
ノード数 （サーバ数）	140 ノード	1520 ノード

本日のプログラム

1. システムのご紹介
2. 利用方法の解説
   - 2-1 システムへの接続
   - 2-2 プログラムの作成・環境設定・コンパイル
   - 2-3 ジョブスクリプトの作成
   - 2-4 ジョブスクリプトの投入
3. 利用を希望する方へ

スーパーコンピュータ利用の流れ

ユーザー端末

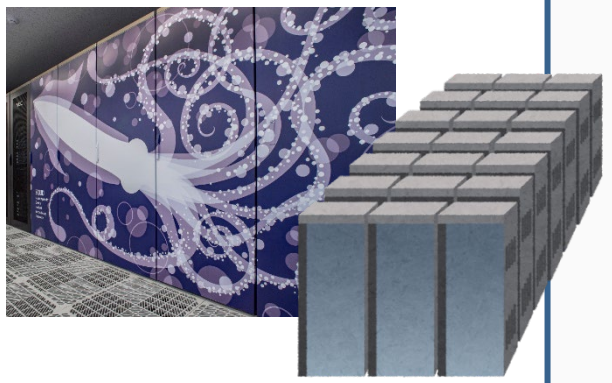
フロントエンド
ノードへの接続

スパコンの
フロントエンドノード
（作業用サーバ）

プログラム準備

ジョブスクリプト作成

スパコンの
計算ノード
（計算用サーバ）



ジョブスクリプト投入

プログラム実行

フロントエンドノードへの接続

SSH (Secure Shell)接続

ターミナル(Mac/Linux)やコマンドプロンプト(Win)を使用
ターミナルソフトを使用(TeraTerm, Putty等)

接続先

[SQUID]　squidhpc.hpc.cmc.osaka-u.ac.jp
[OCTOPUS]　octopus.hpc.osaka-u.ac.jp

接続コマンド例

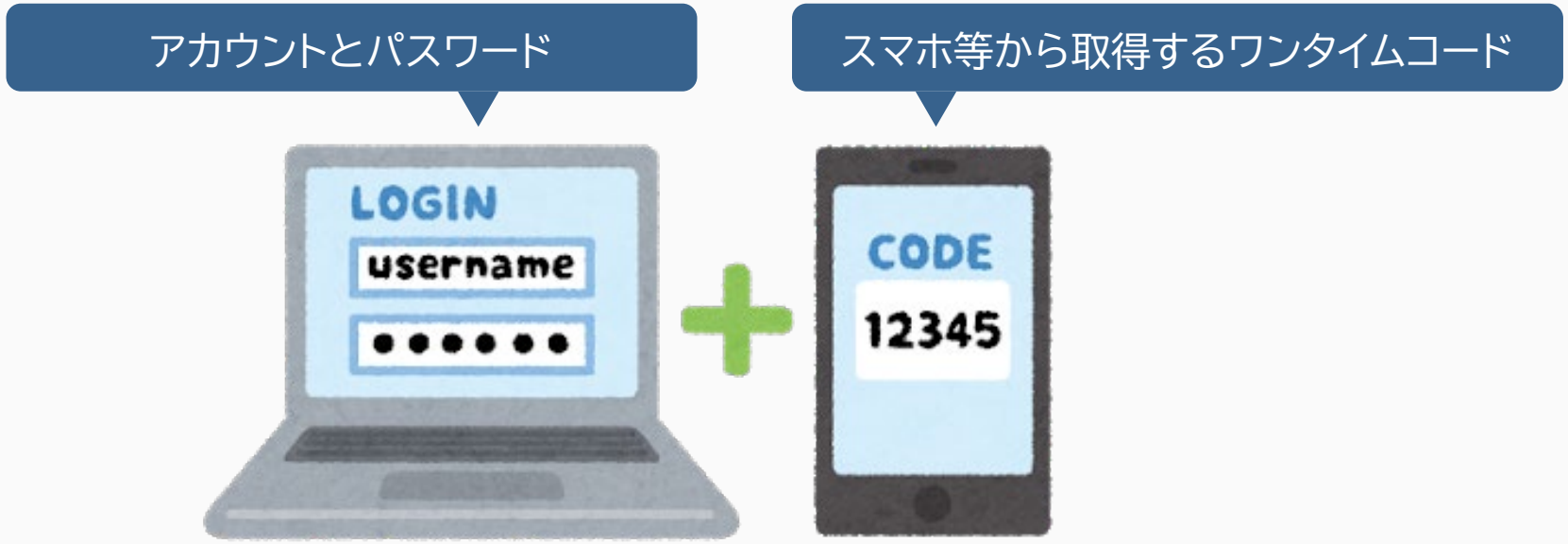
ssh　ユーザアカウント@squidhpc.hpc.cmc.osaka-u.ac.jp

学内/外、国内/外どこからでも接続可能

フロントエンドノードへの接続

SQUID と OCTOPUS は多要素認証でのログインとなります

多要素認証用の端末が必要です



※公開鍵認証には対応していません

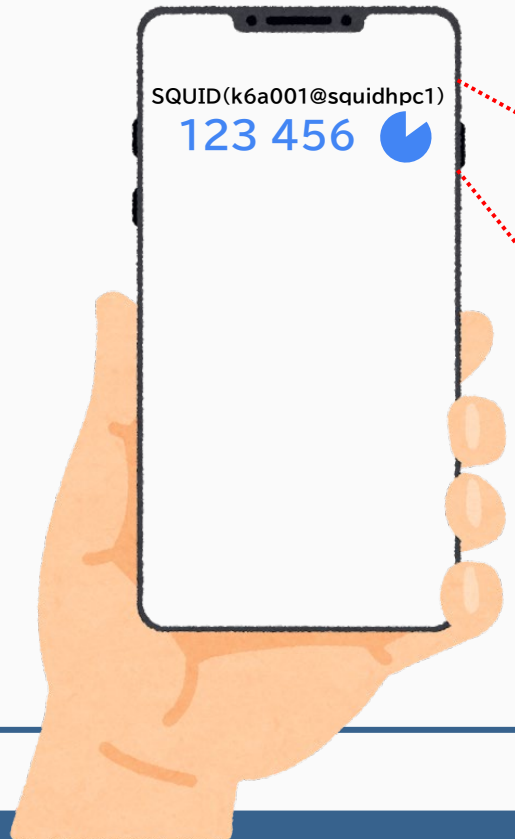
多要素認証用の端末

ご自身のスマートフォンやパソコンを多要素認証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 Microsoft Authenticator	Google Play Store
iOS		Apple App Store
Windows	WinAuth	GitHub
macOS	Step Two	Apple App Store

フロントエンドノードへの接続

スパコン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多要素認証の登録が完了します



Initialize google-authenticator

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

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

Your new secret key is: XXXXXXXXXXX

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

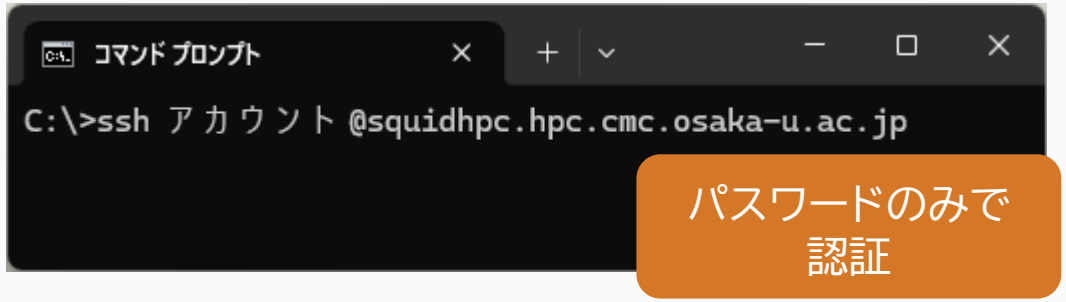
「-1」を入力して
登録完了

[デモ 1] フロントエンドノードへの接続

お持ちのアカウントでスパコンに接続してみましょう！

- 事前に スマートフォン等に 多要素認証のアプリをインストールしてください
- Windowsの方はコマンドプロンプト、Macの方はターミナルを起動してください

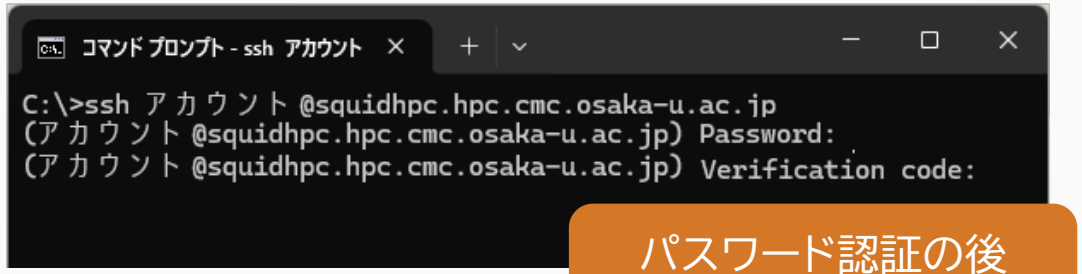
接続 1回目



パスワードのみで認証

→表示されるQRコードを多要素認証アプリで読み込んだ後、「-1」を入力してログアウト

接続 2回目



パスワード認証の後ワンタイムコードを入力

→スパコンのフロントエンドノードに接続完了

スーパーコンピュータ利用の流れ

ユーザー端末

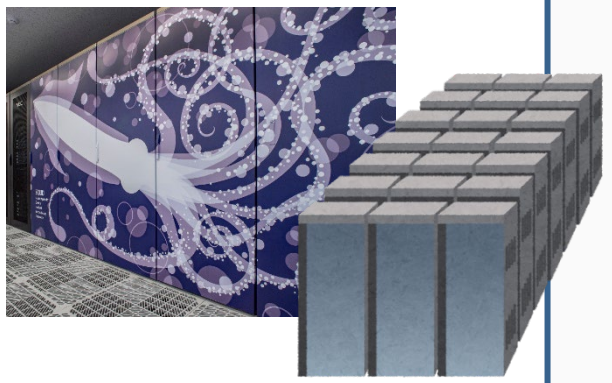
フロントエンド
ノードへの接続

スパコンの
フロントエンドノード
（作業用サーバ）

プログラム準備

ジョブスクリプト作成

スパコンの
計算ノード
（計算用サーバ）



ジョブスクリプト投入

プログラム実行

プログラムの準備

スパコンを利用するために
プログラムやアプリケーションを準備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プログラム言語

Fortran言語、C言語、C++言語、Python、R、Julia

当センターの計算機で使用可能な主なアプリケーション

OpenFOAM、LAMMPS、Gaussian、GROMACS
PyTorch、QuantumESPRESSO、etc…

必要なアプリケーションを
ご自身でインストールすることも可能です！

利用環境の設定

利用するプログラムやアプリケーションに応じて環境の設定が必要

Environment modules というツールを使用

	Intelコンパイラ	NVIDIA HPC SDK	ベクトルコンパイラ	GNUコンパイラ
モジュール	BaseCPU	BaseGPU	BaseVEC	BaseGCC

などなど…

コマンド例

```
module load BaseCPU
  →Intelコンパイラのコマンド「ifx」が使用可能になる

module load BaseApp
module load gromacs
  →アプリケーション GROMACSが使用可能になる
```

プログラムの準備:まとめ

スパコンを利用するために、プログラムやアプリケーションを準備する必要があります

1. 開発した C言語や Fortran言語のプログラムをお持ちの方

→スパコンにプログラムを持ってきて、コンパイルしましょう

2. Python で機械学習をしている方

→スパコンで機械学習のフレームワーク等、Pythonパッケージを準備しましょう

3. オープンソースのアプリケーションで計算されている方

→スパコンに入力ファイル等必要なデータを持ってきましょう

→スパコンにアプリケーションをインストールしましょう

[デモ 2] プログラムの準備

フロントエンドノードでプログラムをコンパイルしてみましょう！

1. サンプルプログラムをコピー
   ※文字入力時は [Tab]キーでの補完機能を活用してください

   $ cp /system/lecture/nyumon/sample.f ~/

2. 汎用CPUノード の環境設定を読み込み

   $ module load BaseCPU

3. Fortran で書かれた sample.f を汎用CPUノード用にコンパイル
   （実行できる形式へ変換する）

   $ ifx sample.f

   → lsコマンドで確認し、「a.out」という実行ファイルが生成されていれば成功です！

スーパーコンピュータ利用の流れ

ユーザー端末

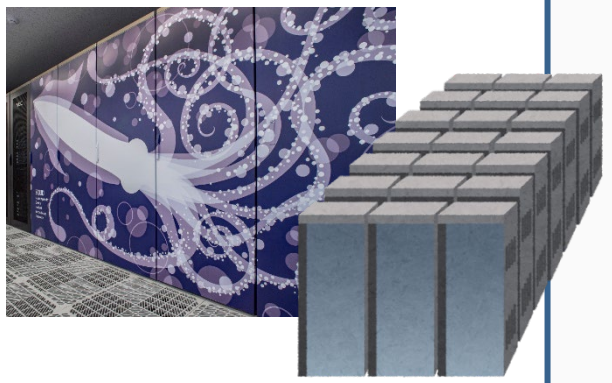
フロントエンド
ノードへの接続

スパコンの
フロントエンドノード
（作業用サーバ）

プログラム準備

ジョブスクリプト作成

スパコンの
計算ノード
（計算用サーバ）



ジョブスクリプト投入

プログラム実行

コンピュータの利用方法

インタラクティブ利用

- 普段のパソコン操作
- コマンドを入力してその場で実行・結果確認する使い方
- 手軽で、短い処理や動作確認に向いている

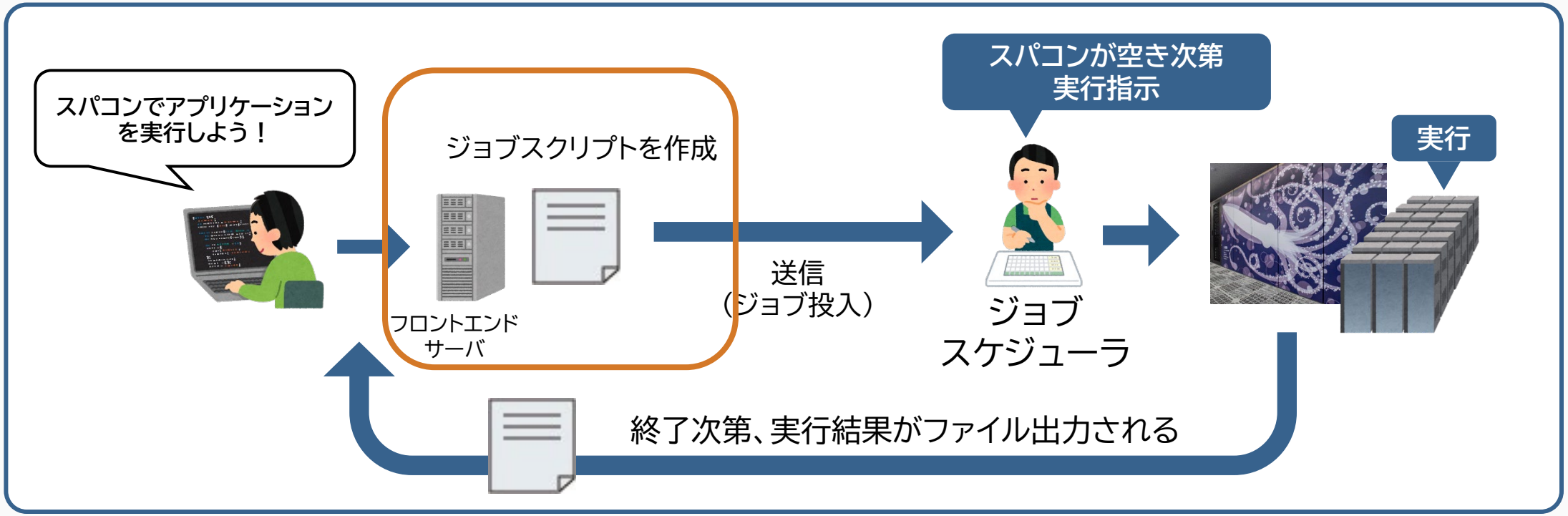
バッチ利用

- 処理内容をまとめて登録し、順番が来たら自動で実行する使い方
- 登録後はログアウトしても実行されるため、長時間の計算に向いている

スパコンはこちら！

バッチ利用

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スパコンが処理を実行



ジョブスクリプトの例

```
#!/bin/bash

#PBS -q SQUID
#PBS -l elapstim_req=1:00:00
#PBS --group=G12345

module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

スパコンのリソースや環境設定、実行する処理を記載する

「**#PBSから始まる行**」と「**シェルスクリプト**」の二段構成

ジョブスクリプトの例：#PBSの行

```
#!/bin/bash
#PBS -q SQUID
#PBS -l elapstim_req=1:00:00
#PBS --group=G12345
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

使用する
リソースや環境

#PBSから始まる行ではスパコンのリソースや環境設定を行う

オプション	説明
#PBS -q	ジョブクラスを指定し、計算に使用する優先度等を指定する
#PBS --group	グループ名の指定
#PBS -l	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CPU数
#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(IntelMPIの場合、#PBS -T intmpi と指定)

ジョブスクリプトの例:ジョブクラス

```
#!/bin/bash
#PBS -q SQUID
#PBS -l elapstim_req=1:00:00
#PBS --group=G12345

module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

使用する
リソースや環境

SQUIDのジョブクラス	利用可能経過時間	利用可能最大コア数	同時利用可能ノード数	備考
SQUID	120時間	38,912 Core (76Core × 512ノード)	512 ノード	
SQUID-H	120時間	38,912 Core (76Core × 512ノード)	512 ノード	高優先度
SQUID-S	120時間	38 Core (76Core × 0.5ノード)	0.5 ノード	ノード共有
DBG	10 分	152 Core (76Core × 2ノード)	2 ノード	デバッグ用

OCTOPUSのジョブクラス	利用可能経過時間	利用可能最大コア数	同時利用可能ノード数	備考
OCT	120時間	32,768 Core (256Core × 128 ノード)	128 ノード	
OCT-H	120時間	32,768 Core (256Core × 128 ノード)	128 ノード	高優先度
OCT-S	120時間	128 Core (256Core × 0.5ノード)	0.5 ノード	ノード共有
DBG	10 分	512 Core (256Core × 2ノード)	2 ノード	デバッグ用

ジョブスクリプトの例:シェルスクリプト

```
#!/bin/bash

#PBS -q SQUID
#PBS -l elapstim_req=1:00:00
#PBS --group=G12345

module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

スパコンで
実行する処理

シェルスクリプトの書式でファイルやディレクトリの実行・操作を記述

利用するプログラムやアプリケーションに応じて環境設定が必要なので
必要に応じて module load を実施してください

環境変数 $PBS_O_WORKDIR には ジョブ投入時のディレクトリが設定される

ジョブスクリプトの例:まとめ

```
#!/bin/bash
#PBS -q SQUID
#PBS -l elapstim_req=1:00:00
#PBS --group=G12345
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

ジョブクラスの指定

リソースの指定
（計算時間とグループ）

スパコンで実行する処理

ジョブ投入時のディレクトリ
に移動して「a.out」を実行

[デモ 3] ジョブスクリプトの作成

ジョブスクリプトを準備しよう！

1. 演習用スクリプト「jobscript.sh」をコピー

   $ cp /system/lecture/nyumon/jobscript.sh ~/

2. jobscript.shを書き換えて汎用CPUノード用のジョブスクリプトを作成

   $ vi jobscript.sh （vi エディタの場合）

   $ emacs jobscript.sh -nw （emacs エディタの場合）

※グループ名は kousyuXXX です。(XXXはアカウントの下3桁)
アカウント:k6b**001** → グループ名:**kousyu001**

※ id コマンドでもグループ名を確認できます。
uid=18XX(k6b001) gid=22000(ocean) groups=22000(ocean),14465(**kousyu001**)

スーパーコンピュータ利用の流れ

ユーザー端末

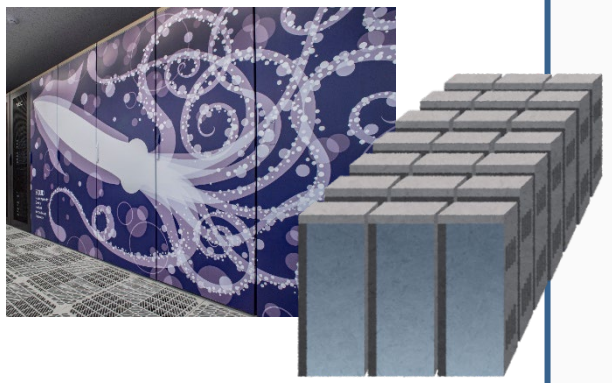
フロントエンド
ノードへの接続

スパコンの
フロントエンドノード
（作業用サーバ）

プログラム準備

ジョブスクリプト作成

スパコンの
計算ノード
（計算用サーバ）

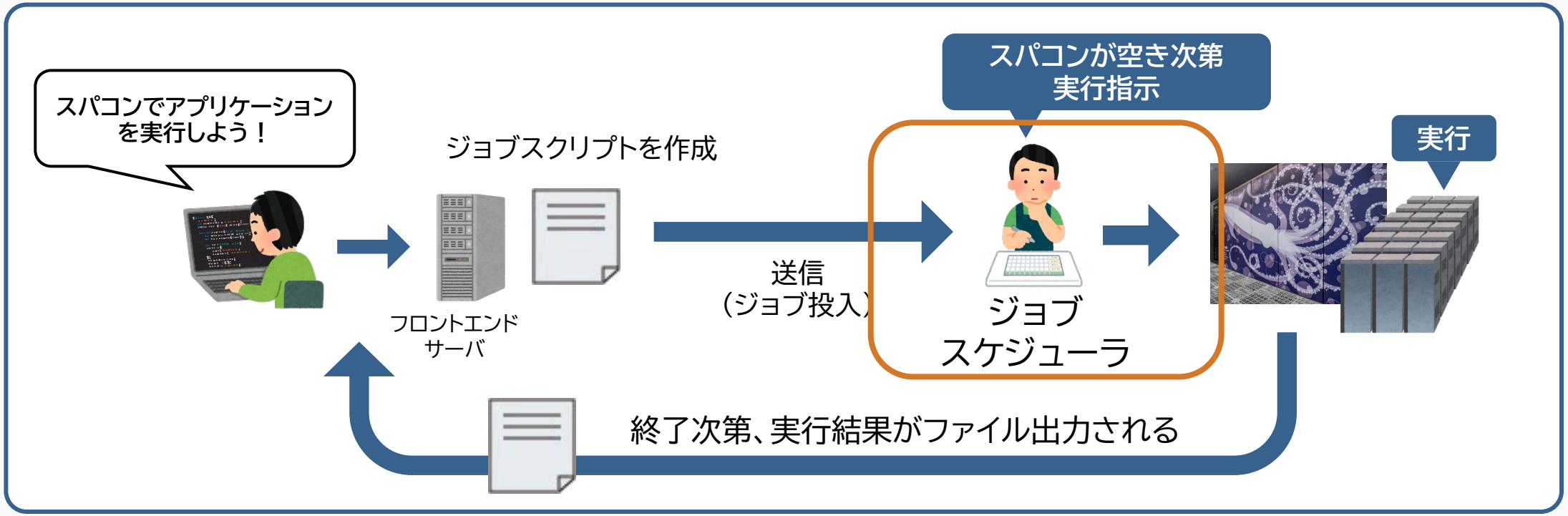


ジョブスクリプト投入

プログラム実行

バッチ利用

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スパコンが処理を実行



ジョブスケジューラとは

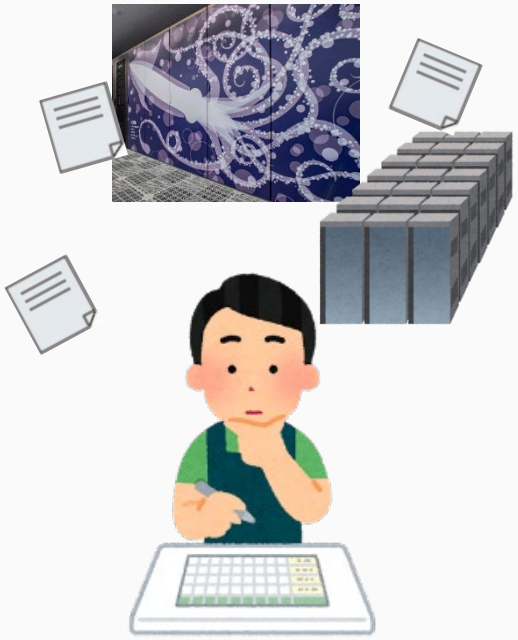
あらかじめ管理者によって設定された割当ポリシーに従い、
ジョブを計算ノードに割り当てるソフトウェア

主な役割

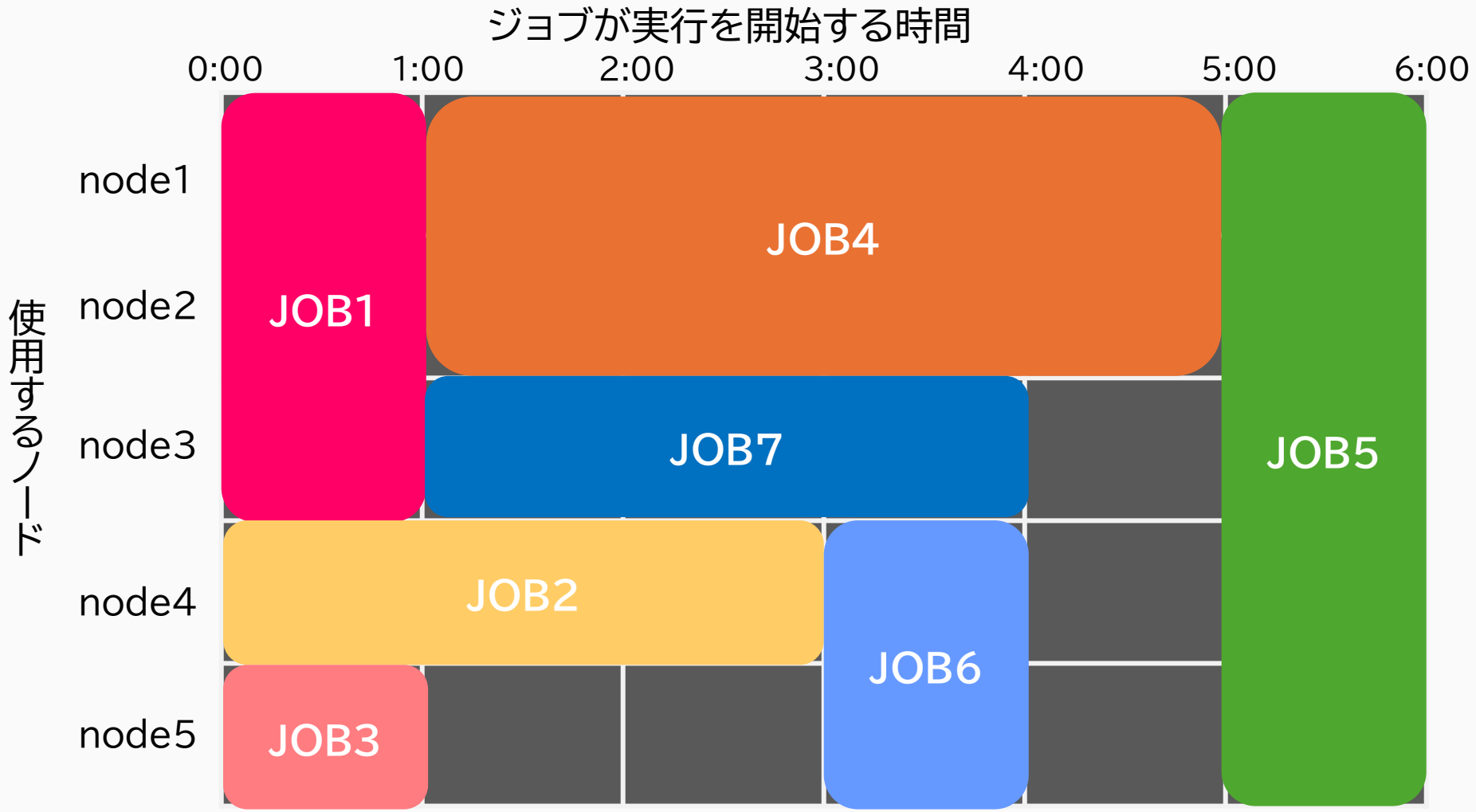
- スパコンの各ノードのCPU/メモリ使用率等を監視、管理
- ジョブスクリプトを受信し、適切なノードを選定
- ジョブ実行に伴う入出力データのファイル転送

当センターのスパコンは **バックフィル型のスケジューラ** を採用

- **数日先まで「ジョブの実行計画表」を作成する**
- 計画表に載れば、そのジョブの実行開始時間が保障される
- ジョブの実行中はリソースを占有して割り当てる

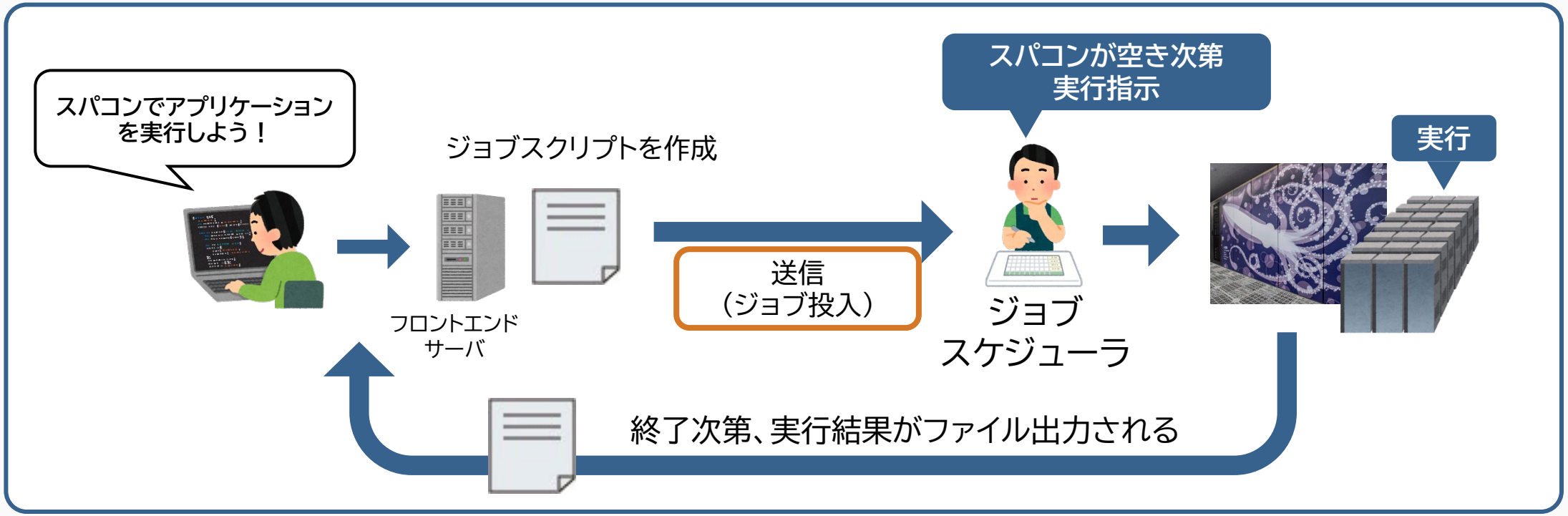


ジョブスケジューラのイメージ



バッチ利用

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スパコンが処理を実行



ジョブの送信方法

フロントエンドノードからジョブスクリプトを送信

コマンド

qsub［ジョブスクリプトファイル］

実行結果

送信に成功すると以下のメッセージが表示されます

Request［リクエストID］submitted to queue:［ジョブクラス名］.

ジョブごとに「リクエストID」という通し番号が付与されます
リクエストIDの例：123456.sqd, 789100.oct

システムで障害発生した場合の連絡に使用する場合があります

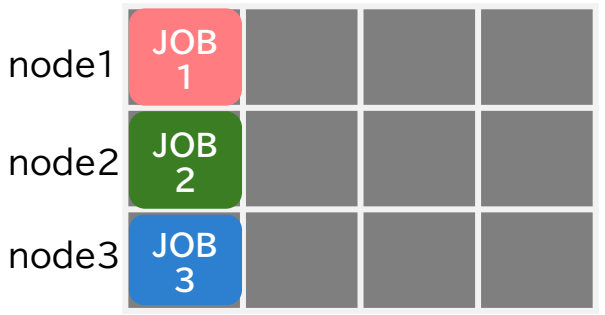
ジョブの投入方法

複数のジョブスクリプトを送信するには？

順不同で実行する場合

コマンド

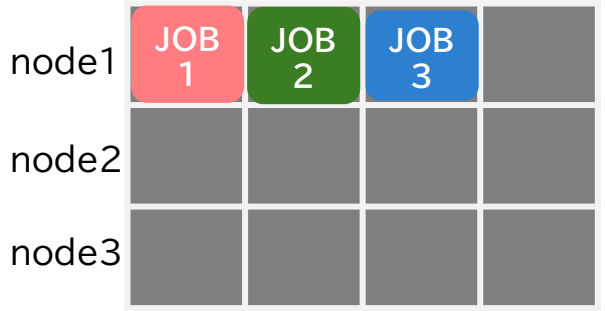
```
$ qsub [JobScript1]
$ qsub [JobScript2]
$ qsub [JobScript3]
```



順番通りに実行する場合

コマンド

```
$ qsub [JobScript1] [JobScript2] [JobScript3]
```



投入済みジョブの確認方法（1）

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（2）

ジョブの実行開始時刻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	Date(PLANNED START)
1234.sqd	job-test	k6a001	SQUID	0	PRR	2026-05-20 18:00:53

実行開始予定時刻

他の方のジョブが早く終わると前倒しで実行が始まる場合があります。
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。

実行開始までは「sstat」コマンド → 実行開始後は「qstat」コマンド

投入済みジョブの削除方法

ジョブの実行キャンセルが可能

コマンド

qdel [リクエストID]

実行結果

キャンセルに成功すると
“Request [リクエストID] was deleted”と表示されます

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

実行結果：ジョブスクリプト名.oリクエストID
実行エラー：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ます

catやlessコマンドでファイルの内容を出力し確認

コマンド

```
cat jobscript.sh.o123456
```

意図通りの結果が表示されていれば計算は成功です！

プログラムの実行：まとめ

スパコンでプログラムやアプリケーションを実行する際は「バッチ利用」

1. ジョブスクリプトを作成する
→右のようなファイルを作成

```
#!/bin/bash

#PBS -q SQUID
#PBS -l elapstim_req=1:00:00
#PBS --group=G12345

module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

2. ジョブスクリプトをSQUIDに送信する
→qsubコマンドを使用

3. 定期的にジョブの状態を確認する
→qstatやsstatコマンドを使用

4. 実行終了したら結果を確認する
→フロントエンドノード上に出力されたファイルを開いて確認

［最終デモ］ジョブの実行

スパコンでジョブを実行して結果を確認しましょう！

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 jobscript.sh

リクエストIDを確認しておきましょう

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 jobscript.sh.oXXXXX（実行結果）
   $ cat jobscript.sh.eXXXXX（実行エラー）

サンプルプログラムを実行した場合は 実行結果ファイルに
「Hello, World!」が出力されます

実行エラーファイルには
Loading compiler version 2023.2.4…
Loading BaseCPU/2026…
等のメッセージが表示されますが これは
module load BaseCPU の結果ですのでエラーではありません

より発展的な利用に向けて

利用の参考になるWebページ

D3センター 大規模計算機システム Webページ
https://www.hpc.cmc.osaka-u.ac.jp

利用方法
https://www.hpc.cmc.osaka-u.ac.jp/system/manual/
FAQ
https://www.hpc.cmc.osaka-u.ac.jp/faq/
問い合わせフォーム
https://www.hpc.cmc.osaka-u.ac.jp/support/contact/auto form/
研究成果
https://www.hpc.cmc.osaka-u.ac.jp/researchlist/

より発展的な利用に向けて

本日以降の講習会・セミナー（全てオンライン）

開催日	講習会名	概要
6/3	スパコンに通じる並列プログラミングの基礎	並列プログラミングの手法や考え方の基礎
6/5	OpenMP入門	OpenMPによる一般的な並列プログラミングの基礎とその利用方法 【お試しアカウント付き】
6/10	スーパーコンピュータ バッチシステム入門 / 応用	計算機利用（バッチ利用）の概要 【お試しアカウント付き】
6/12	コンテナ入門	Singularityを利用したコンテナジョブの投入方法と コンテナのカスタマイズ方法を説明【お試しアカウント付き】
6/16	SX-Aurora TSUBASA 高速化技法の基礎	SQUID ベクトルノード群で動作するプログラムの高速化を目的とし 性能測定や基礎的なチューニング手法を説明【お試しアカウント付き】
6/24	並列プログラミング入門（OpenMP/MPI）	OpenMP、MPI、自動並列化機能による 並列プログラミングの基礎と利用方法を説明【お試しアカウント付き】
7/15	汎用CPUノード 高速化技法の基礎	Intelコンパイラの基礎的なチューニング手法の説明【お試しアカウント付き】
調整中	GPUプログラミング入門（OpenACC）	GPUおよびGPUを用いた計算の特徴や概要を説明【お試しアカウント付き】
	GPUプログラミング実践（OpenACC）	OpenACCによる応用的なGPUプログラミングを説明【お試しアカウント付き】

詳細はWEBをご参照ください　https://www.hpc.cmc.osaka-u.ac.jp/lecture_event/

本日のプログラム

1. システムのご紹介
2. 利用方法の解説
   - 2-1 システムへの接続
   - 2-2 プログラムの作成・環境設定・コンパイル
   - 2-3 ジョブスクリプトの作成
   - 2-4 ジョブスクリプトの投入
3. 利用を希望する方へ

利用を希望する方へ

本センターのスパコン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（利用料）が必要になります！

利用負担金制度

一般利用（学術利用）

産業利用
成果公開型

産業利用
成果非公開型

SQUID

OCTOPUS
Osaka university Compute & sTOrage Platform Urging open Science

共有利用	
10万円+税	1,000 ポイント
50万円+税	5,250 ポイント
100万円+税	11,000 ポイント
300万円+税	34,500 ポイント
500万円+税	60,000 ポイント

占有利用	
1,150,000 円+税	SQUID 汎用CPU 1ノード/年

※SQUIDとOCTOPUS 両方を使用する場合は
個別にお申し込みが必要です

＋

HDDストレージ
初期容量5TB
2,000円/TB で追加可能

SSDストレージ(SQUIDのみ)
初期容量なし
5,000円/TB で追加可能

料金
5倍

詳細はWEBページをご参照ください https://www.hpc.cmc.osaka-u.ac.jp/service/cost/

スパコンの提供方法

共有利用

「OCTOPUSポイント」あるいは
「SQUIDポイント」単位で利用

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

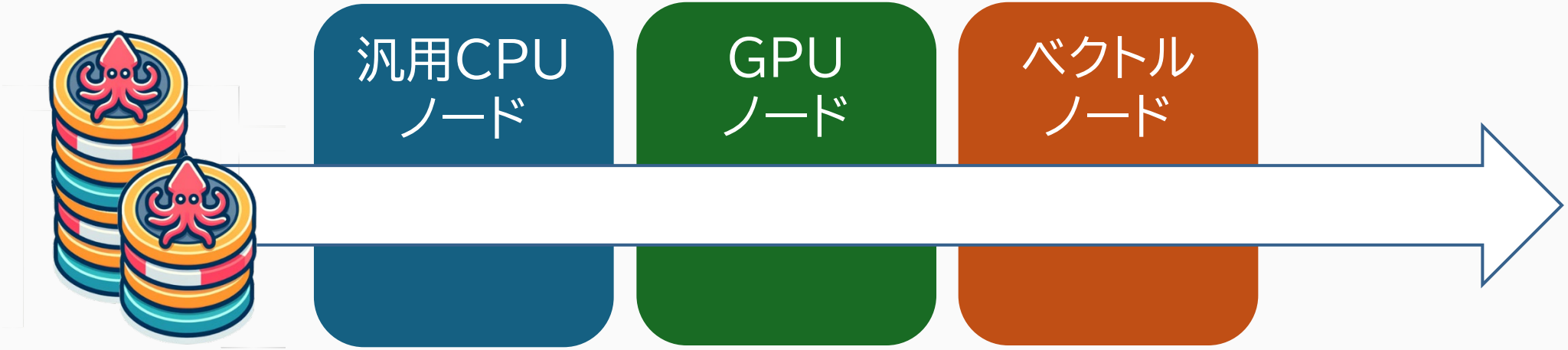
他のグループとノードを共有しない

大規模なノード間並列は試しにくい
原則「待ち時間」が発生しない

「SQUID / OCTOPUSポイント」とは

計算ノードの使用時間とノード数に応じて消費されるポイント

- SQUIDでは 3つのノード群を横断的に使用可能
- 同じ計算時間でもノード群や優先度に応じて消費量が異なる
- OCTOPUSポイントとSQUIDポイントの交換はできません



「SQUID / OCTOPUSポイント」とは

ポイントの消費量は以下の計算式から算出されます

使用したノード時間 × 消費係数 × 季節係数 × 燃料係数

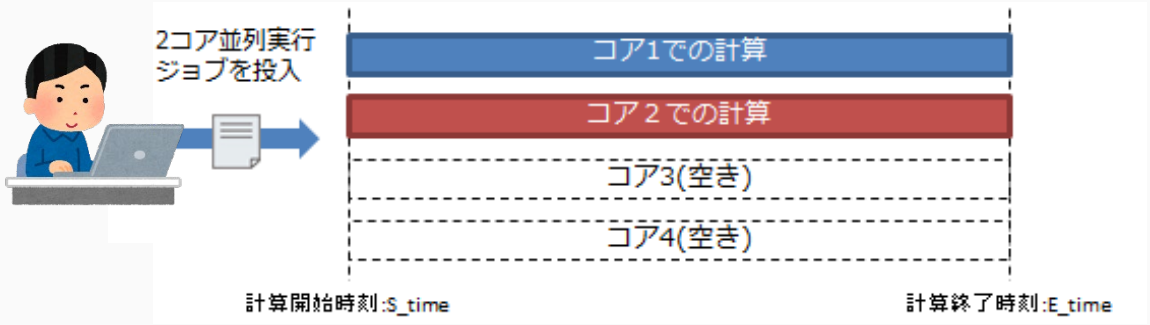
「ノード時間」とは

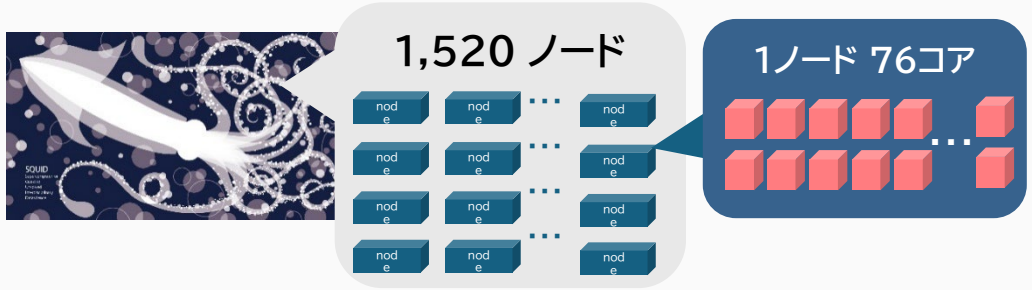
ノード時間 ＝
　計算に使用するノード数 × 計算時間(単位:時間)

(例)

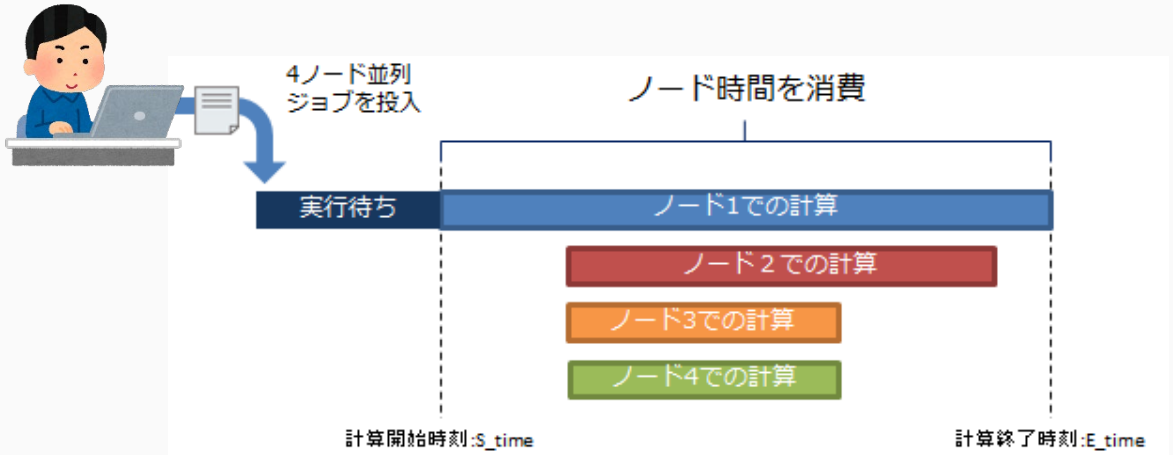
1ノードで3時間の計算	→	3ノード時間消費
30ノードで5時間の計算	→	150ノード時間消費
100ノードで1時間の計算	→	100ノード時間消費
1ノードで100時間の計算	→	100ノード時間消費

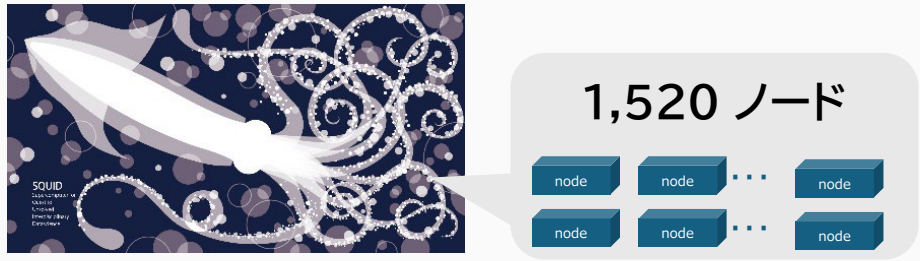
「ノード時間」とは





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

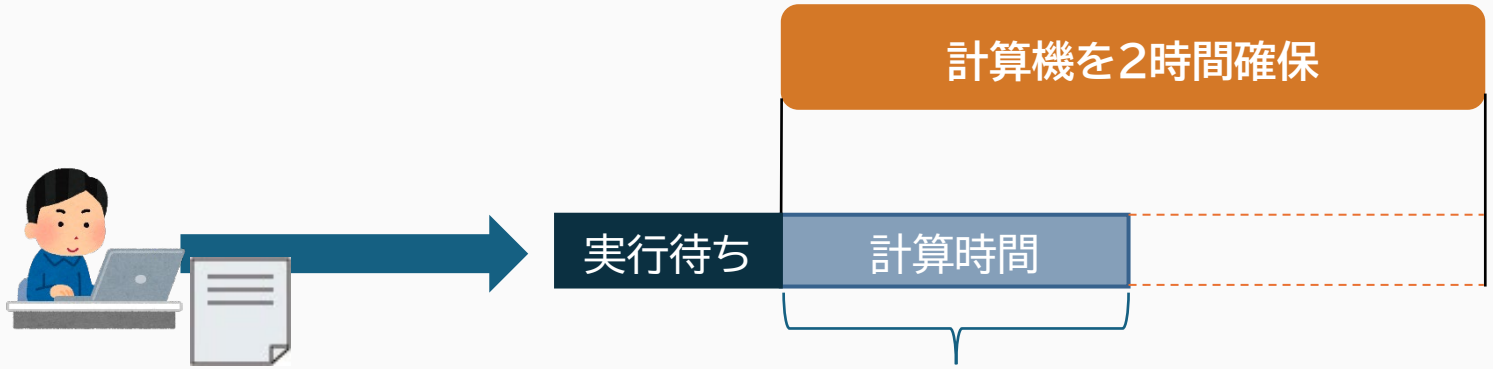




ノード時間は 4ノード ×（計算終了時間 - 計算開始時間）です

「ノード時間」とは

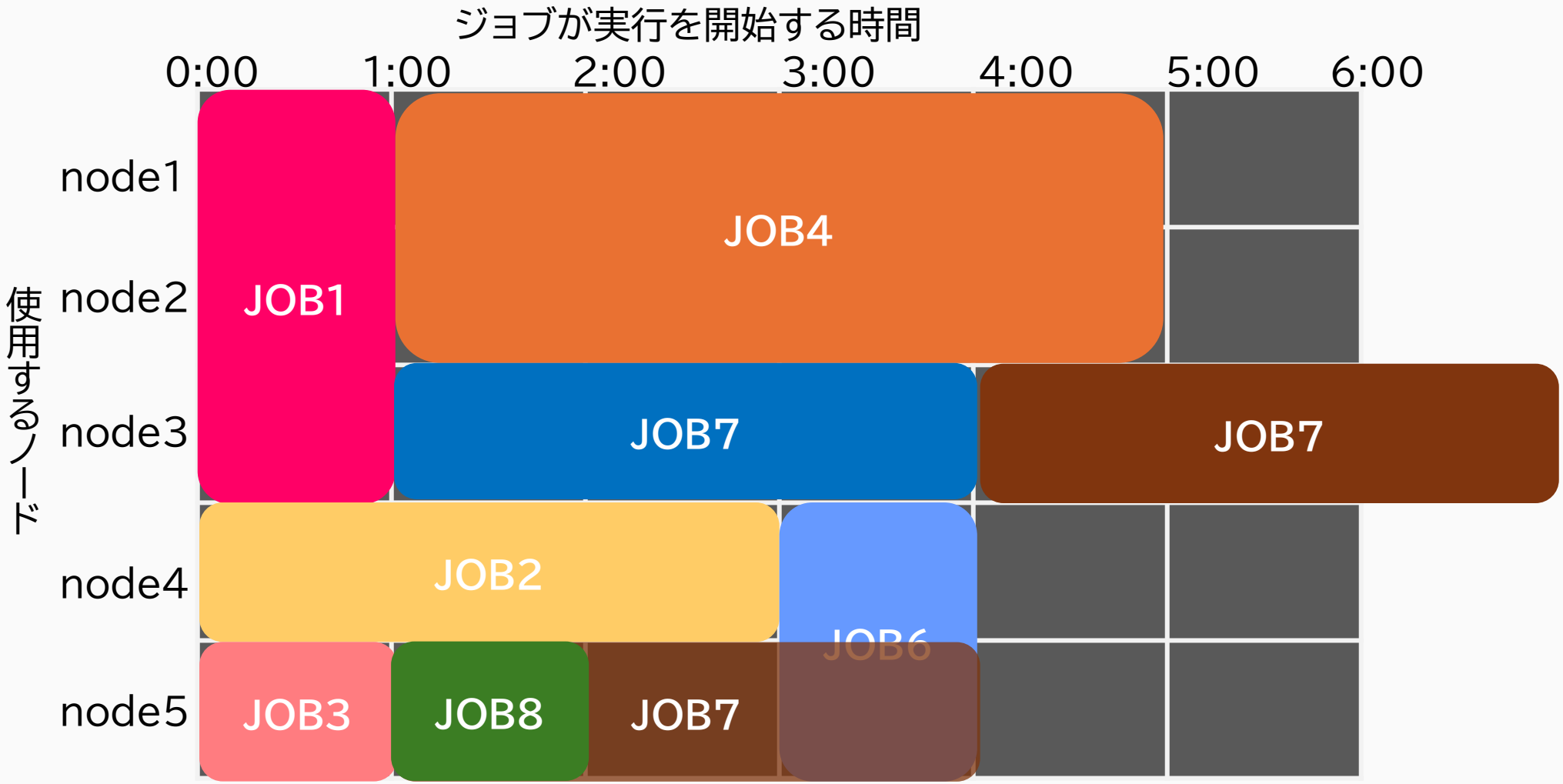
```
#!/bin/bash
#PBS -q SQUID
#PBS -l elapstim_req=2:00:00
```



実際には1時間で終了 → 消費は1時間

→消費するノード時間は、実際にかかった計算時間のみ

スケジューラのイメージ



「消費係数」について

使用したノード時間 × **消費係数** × 季節係数 × 燃料係数

消費係数

SQUID	高優先度	通常優先度	シェア
汎用CPUノード群	0.3746	0.2998	0.2248
GPUノード群	2.2934	1.8348	1.3762
ベクトルノード群	1.4140	1.1312	0.848

OCTOPUS	高優先度	通常優先度	シェア (1/2)	シェア (1/4)
汎用CPUノード群	0.6219	0.5287	0.3732	0.2488

同じノード時間を使用しても、
SQUIDポイントの消費量は異なる

「季節係数」「燃料係数」について

使用したノード時間 × 消費係数 × **季節係数** × **燃料係数**

季節係数

前年度の利用率を元に
0を超える1 以下の値を設定

燃料係数

変動する電気料金に合わせた値を設定

ノード群	季節係数（2026年4月1日～2027年3月31日）				燃料係数
	4-6月	7-9月	10-12月	1-3月	
OCTOPUS 汎用CPUノード群	1.0	1.0	1.0	1.0	0.85（2026年4月時点）
SQUID 汎用CPUノード群	1.0	1.0	1.0	1.0	
SQUID GPUノード群	1.0	1.0	1.0	1.0	
SQUIDベクトルノード群	1.0	1.0	1.0	1.0	

（例）2025年度4月～6月の利用率が低い
→2026年度4月～6月の季節係数を低く設定

（例）電気料金が値下げ
→燃料係数を0.85に設定

「SQUID / OCTOPUSポイント」について

使用したノード時間 × 消費係数 × 季節係数 × 燃料係数

消費係数

SQUID	高優先度	通常優先度	シェア
汎用CPUノード群	0.3746	0.2998	0.2248
GPUノード群	2.2934	1.8348	1.3762
ベクトルノード群	1.4140	1.1312	0.848

OCTOPUS	高優先度	通常優先度	シェア(1/2)	シェア(1/4)
汎用CPUノード群	0.6219	0.5287	0.3732	0.2488

季節係数・燃料係数

ノード群	季節係数(2026年度)				燃料係数
	4-6月	7-9月	10-12月	1-3月	
OCTOPUS 汎用CPUノード群	全期間「1.0」を設定				0.85 (2026年4月時点)
SQUID 汎用CPUノード群					
SQUID GPUノード群					
SQUIDベクトルノード群					

SQUID 汎用CPUノード群を10ノード並列実行で3時間使用した場合（季節係数:1、燃料係数:0.85）

10 × 3 × 0.2998 × 1 × 0.85 = 7.6449 SQUIDポイントを消費

「SQUID / OCTOPUSポイント」の目安

10万円コース（1000 ポイント）で利用できるノード時間の目安

	消費係数	季節係数	燃料係数	ノード時間の目安
OCTOPUS 汎用CPUノード群	0.5287	1	0.85	2,225 ノード時間
SQUID 汎用CPUノード群	0.2998			3,924 ノード時間
SQUID GPUノード群	1.8348			641 ノード時間
SQUID ベクトルノード群	1.1312			1,040 ノード時間

※通常優先度で実行した場合

まずは試用制度をお試しください

3カ月間 **無料**で以下の資源をご提供

SQUID

75 SQUIDポイント
＋ ストレージ 5 TB

汎用CPU ノード
294 ノード時間

GPU ノード
48 ノード時間

ベクトルノード
78 ノード時間

150 OCTOPUSポイント
＋ ストレージ 5 TB

汎用CPU ノード
294 ノード時間

OCTOPUS
Osaka university Compute & sTOrage Platform Urging open Science

- アプリケーション等 計算環境や技術サポートは有償利用と同等に使用可能
- 有償利用へアカウントの移行も可能

利用申請について

スパコンシステムの利用申請は**随時受け付け中**です！

- 申請方法はWEBをご参照ください
- 通常は申請後、1-2業務日以内にアカウント発行可能です

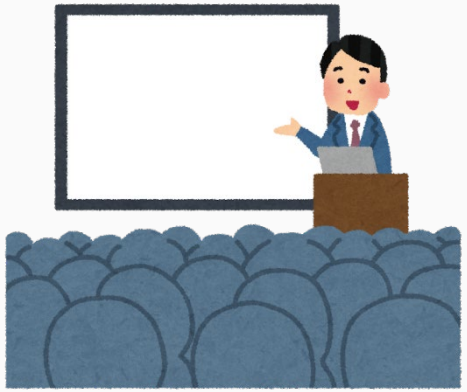
利用は年度単位（4月から翌年3月まで）

- 使いきれなかったノード時間、ポイントは3月末で失効します
- 年度途中でノード時間、ポイントの追加が可能です

利用開始後のサポートについて

WEB

メール・電話



講習会/セミナー

高速化支援

対面利用相談

初めてのスパコン：まとめ

- ご自身で開発したプログラム、オープンソースのアプリケーション等、柔軟に使用可能
- スパコンは「バッチ利用」
  - たくさんの人が同時に、計算規模に応じてスパコンを切り出して使う
  - ジョブスクリプトを使って、スパコンに計算を指示
- 共有利用は「ポイント制」
- スパコンを使ってみたい方は試用制度や各種講習会へ！
- 疑問があれば system@cmc.osaka-u.ac.jp まで！