スパコンの使い方

大阪大学 情報推進部 情報基盤課

利用方法の解説

本講習会では初めてスパコンを使う方を対象に
SQUIDの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も、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

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

SQUID

- 3種類の計算ノードと21PBのストレージで構成される
- 総理論演算性能は16.591 PFLOPS
- 阪大だけでなく国内外の研究者に提供

	CPUノード	GPUノード	ベクトルノード
コア数	76	76	VH:24 VE:80
演算性能	5.837 TFLOPS	161.837 TFLOPS	24.56 TFLOPS
メモリ	256GB	512GB	VH:128GB VE:48GB
ノード数	1520ノード	42ノード (8GPU / ノード)	36ノード (8VE / ノード)

合計1,598ノード　16.591PFLOPS

スーパーコンピュータ利用の流れ

ユーザー端末

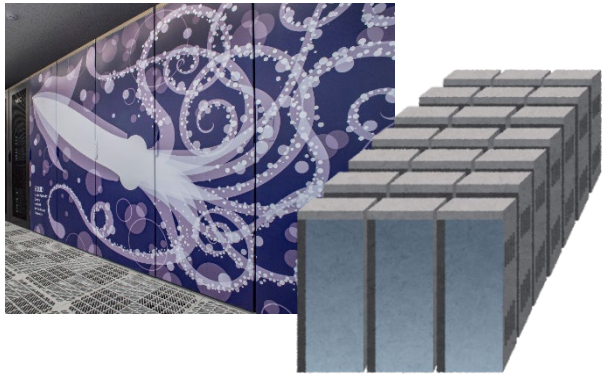
フロントエンド
ノードへの接続

SQUID
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

SQUID
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

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

接続コマンド例

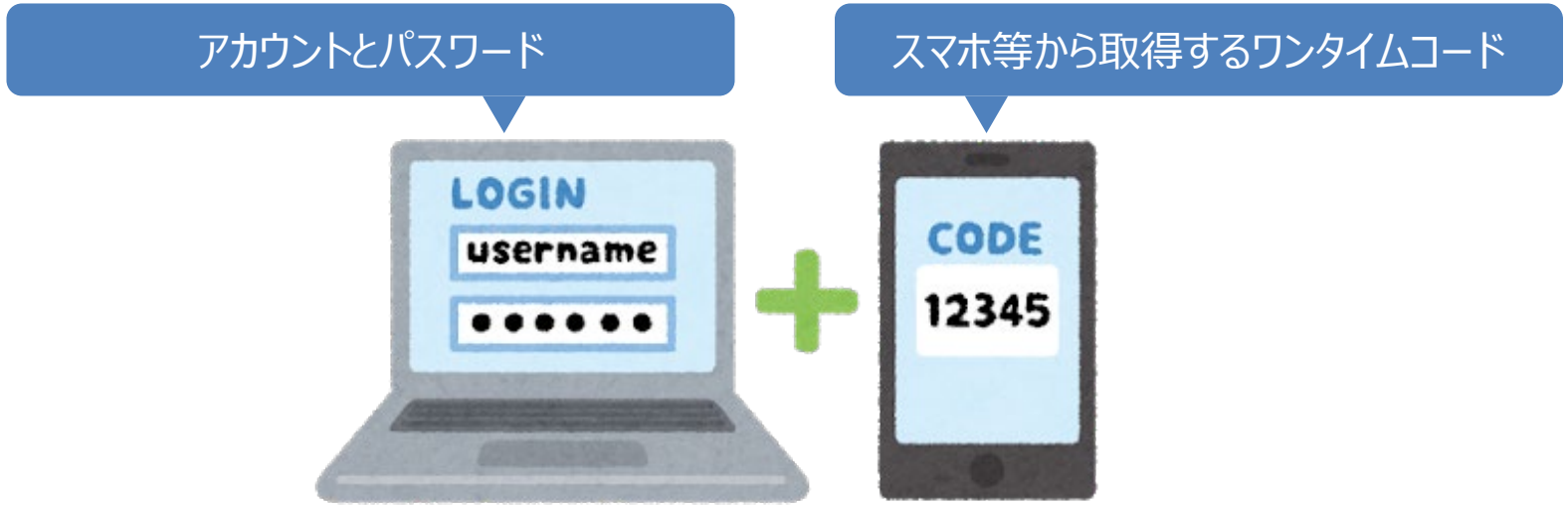
ssh 利用者番号@squidhpc.hpc.cmc.osaka-u.ac.jp

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

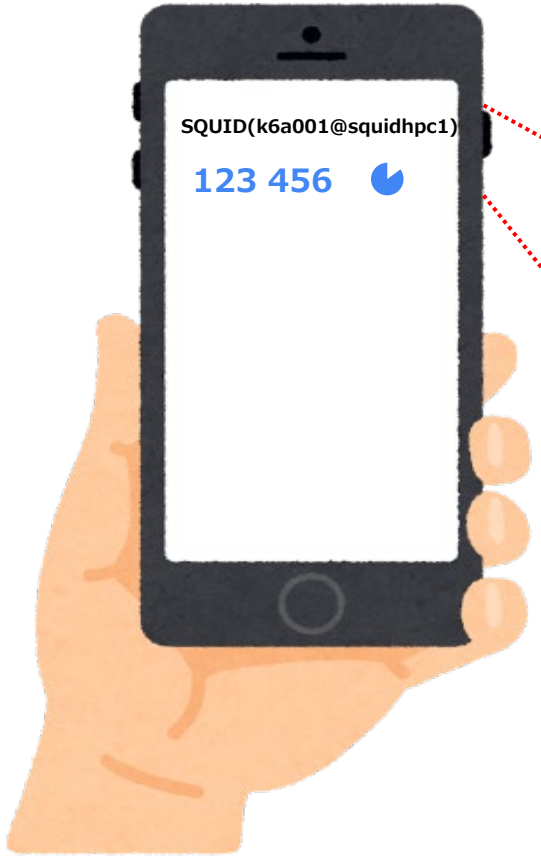
ご自身の**スマートフォン**や**パソコン**を多要素認証用の端末としてお使いください

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 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

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

「-1」を入力して
登録完了

Your new secret key is: XXXXXXXXXX

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

フロントエンドノードへの接続：デモ

お持ちのアカウントでSQUIDに接続します

接続 1回目

$ ssh アカウント@squidhpc.hpc.cmc.osaka-u.ac.jp

→パスワードのみで認証

　表示されるQRコードを読み込んでワンタイムコードを取得し、ログアウト

接続 2回目

$ ssh アカウント@squidhpc.hpc.cmc.osaka-u.ac.jp

→パスワード認証の後、ワンタイムコードを入力

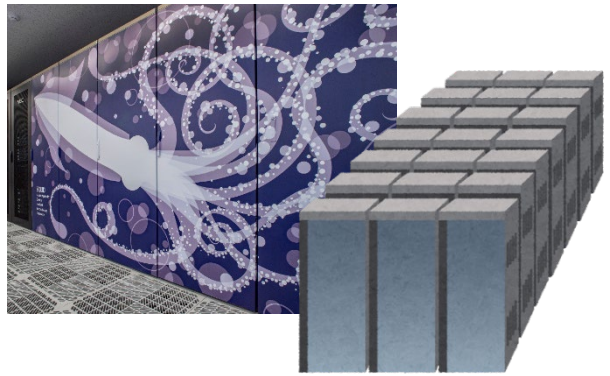
　SQUIDのフロントエンドノードに接続完了

スーパーコンピュータ利用の流れ

ユーザー端末

SQUID
フロントエンドノード
(作業用サーバ)

SQUID
計算ノード
(計算用サーバ)



フロントエンド
ノードへの接続

プログラム準備

ジョブスクリプト
作成

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！

プログラムの準備：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

コマンド例

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

module load BaseApp
module load gromacs/2021.2
  →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

プログラムの準備：デモ

1. サンプルプログラムをコピー
   $ 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)
   $ ifort 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

スーパーコンピュータ利用の流れ

ユーザー端末

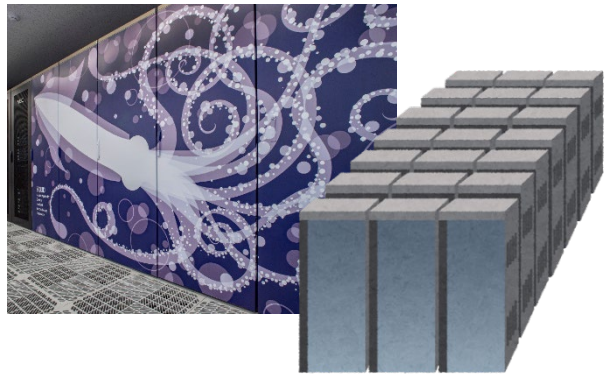
フロントエンド
ノードへの接続

SQUID
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**

SQUID
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

計算機の利用方法

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

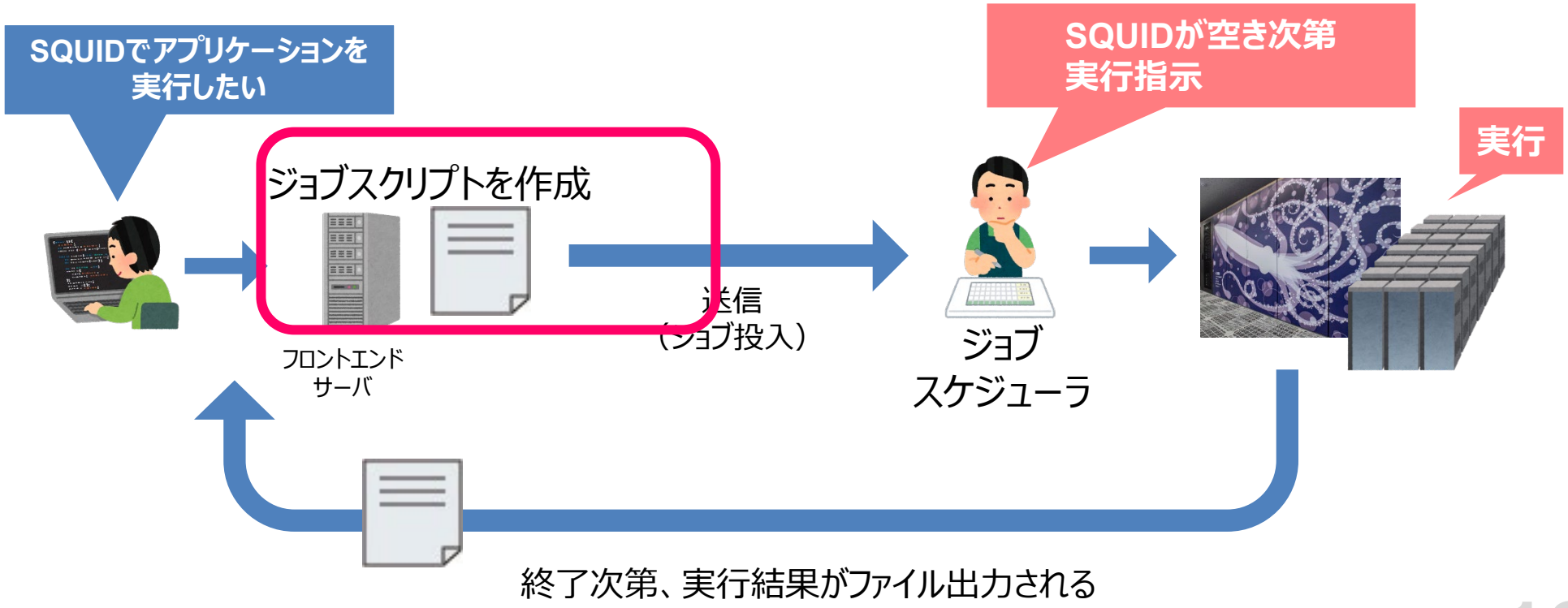
バッチ利用

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

バッチ利用

処理を「ジョブスクリプト」に記述し送信 (→ジョブ)
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

```
#!/bin/bash

#PBS –q SQUID
#PBS –l elapstim_req=1:00:00

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

SQUIDのリソースや環境設定

実行したい処理を記載したシェルスクリプト

ジョブスクリプト

```
#!/bin/bash

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

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

使用する
リソースや環境

NQSオプション(＃PBS～)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、計算に使用する優先度等を指定する**
#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CPU数
#PBS --group	**グループ名の指定**
#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)

必須！

ジョブスクリプト

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

使用する
リソースや環境

ジョブクラス	利用可能 経過時間	利用可能 コア数	同時利用 可能ノード数	備考
SQUID	120時間	38,912Core (76Core × 512ノード)	512ノード	
SQUID-H	120時間	38,912Core (76Core × 512ノード)	512ノード	高優先度
SQUID-S	120時間	38Core (76Core × 0.5ノード)	0.5ノード	ノード共有

ジョブスクリプト

```
#!/bin/bash

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

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

SQUIDで
実行する処理

ファイルやディレクトリの実行・操作を記述(シェルスクリプト)

- 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
  →module loadを実施してください
- $PBS_O_WORKDIR ：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

```
#!/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

環境設定

ジョブ投入時のディレクトリへ移動

a.outを実行する

ジョブスクリプトの作成：デモ

1. 演習用スクリプトをコピー

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

2. jobscript.shを元に汎用CPUノード用のジョブスクリプトを作成

   $ emacs jobscript.sh –nw

（参考）emacsの操作方法： 保存 ctrl-x → ctrl-s
終了 ctrl-x → ctrl-c

※グループ名は kousyuXXX です。（XXXは利用者番号の下３桁）
利用者番号：k6b001 ⇨ グループ名：kousyu001

【参考】自身のグループ名は id コマンドでも確認できます。

グループ名

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

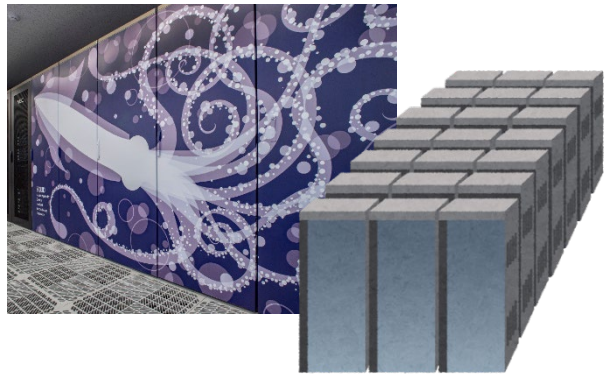
フロントエンド
ノードへの接続

SQUID
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

SQUID
計算ノード
(計算用サーバ)

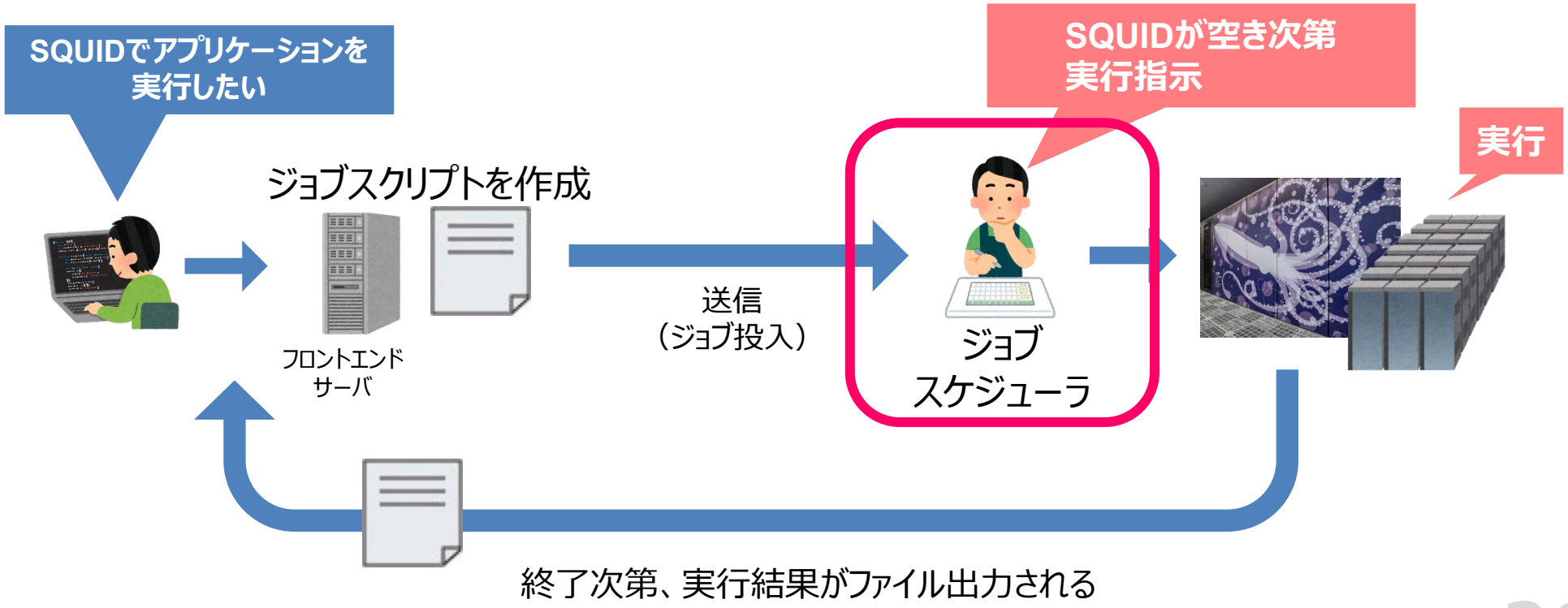


ジョブスクリプト投入

プログラム実行

バッチ利用

処理を「ジョブスクリプト」に記述し送信 (→ジョブ)
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

計算機システム各ノードのストレージ容量、メモリ容量、性能、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

ジョブスケジューラとは

当センターでは**バックフィル型**を採用

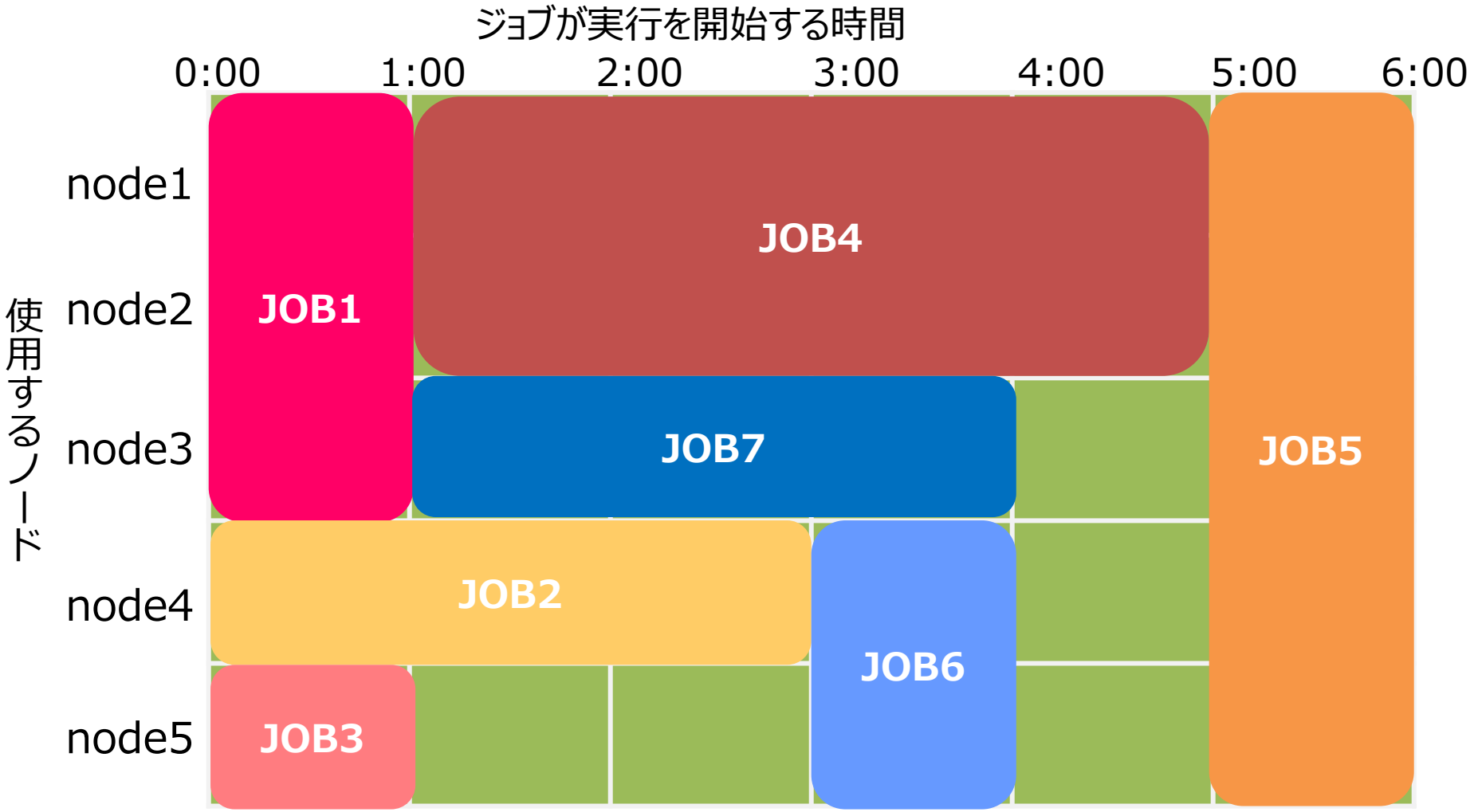
特徴

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

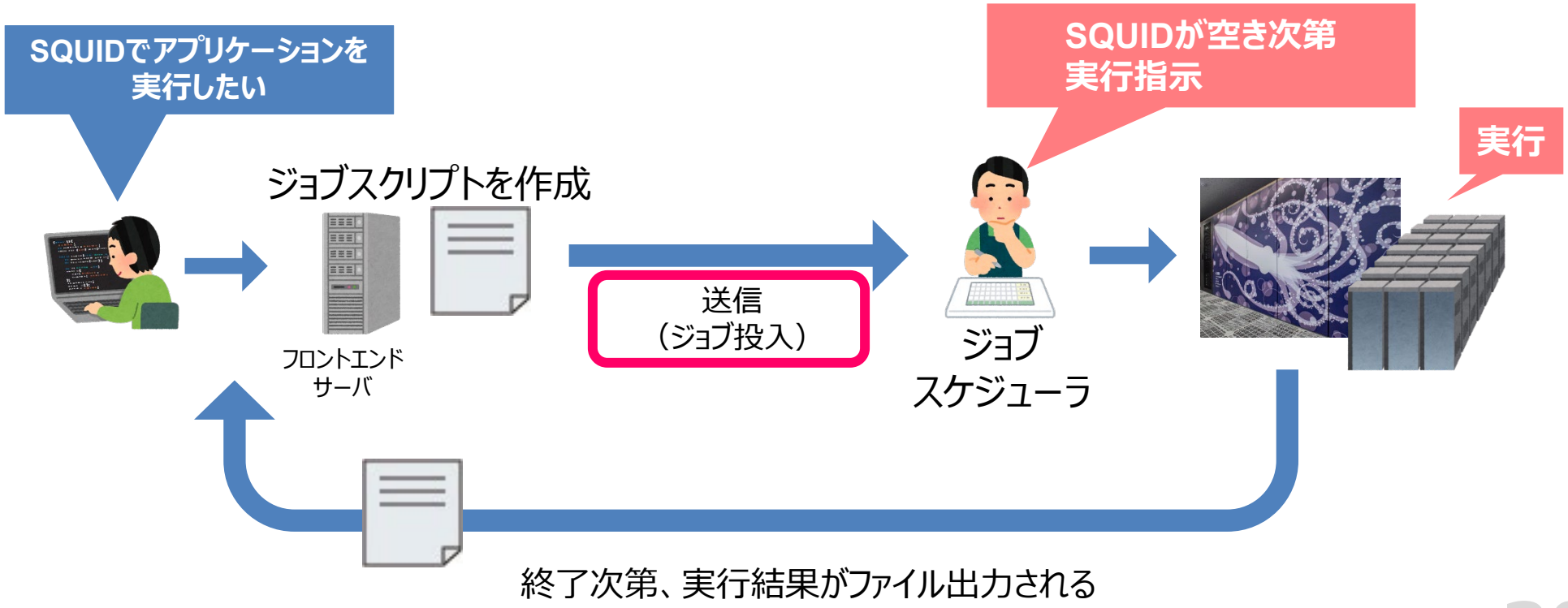
実行中は指定したリソースを占有して割り当てる

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
スクリプトに基づき計算機が処理を実行



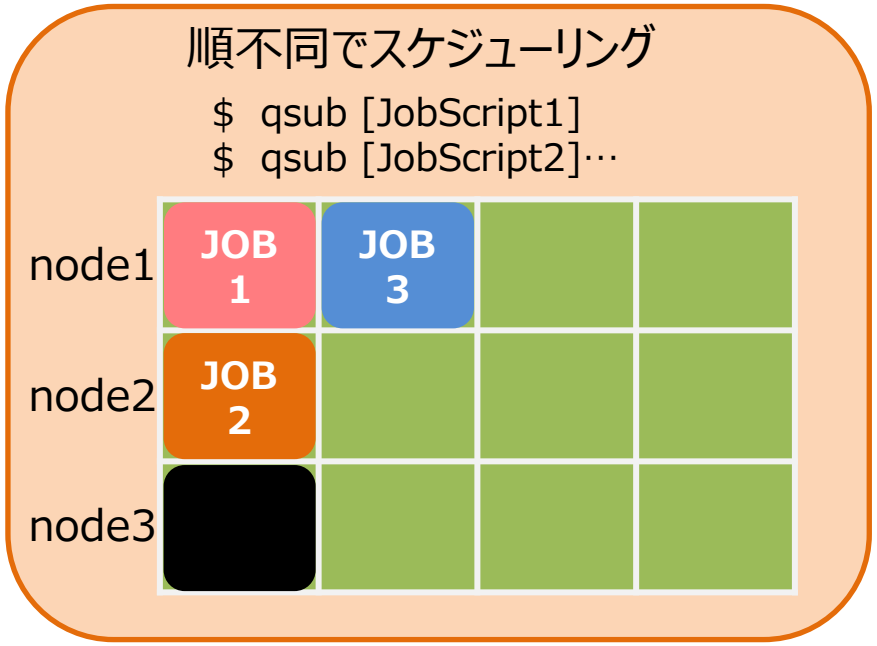
ジョブの投入方法

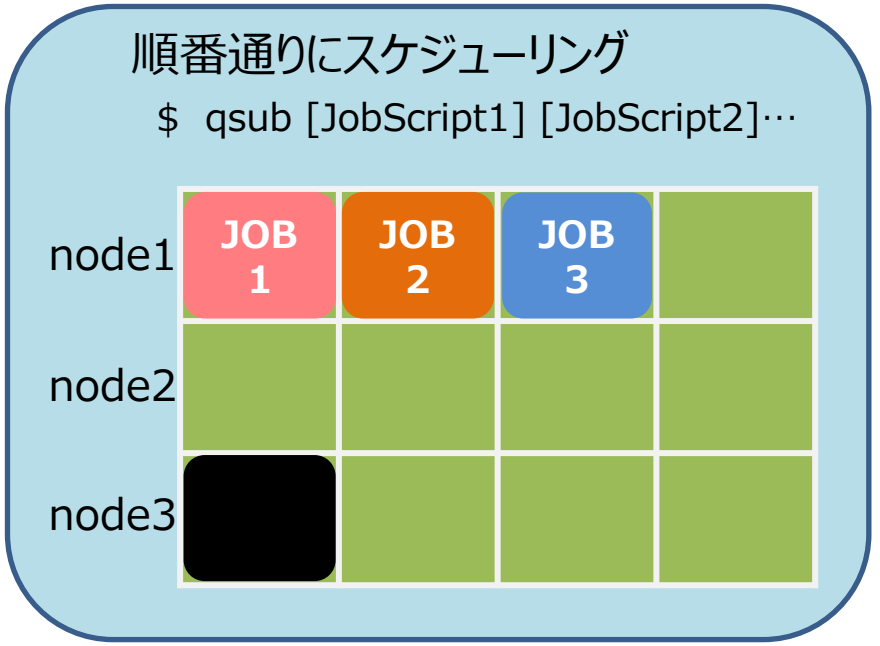
フロントエンド端末からジョブスクリプトを送信

コマンド

$ qsub [ジョブスクリプトファイル]

（参考）複数ジョブを投入する場合





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	2023-09-04 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

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る

catやlessコマンドでファイルの内容を出力し確認

```
$ cat jobscript.nqs.o123456
```

意図通りの結果が表示されていれば計算は成功！

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

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**
→サーバ上に保存されたファイルを開いて確認

```
#!/bin/bash

#PBS –q SQUID
#PBS –l elapstim_req=1:00:00

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

ジョブスクリプトの投入：デモ

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

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
   $ cat jobscript.sh.eXXXXX

より発展的な利用に向けて

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
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
9/2	Pythonチュートリアル（初級編）Day1	Pythonの実行方法、プログラミングの初歩
9/5	初めてのスパコン	スーパーコンピュータの基礎的な知識と、その使い方 初心者向け【お試しアカウント付き】
9/9	Pythonチュートリアル（初級編）Day2	Pythonの実行方法、プログラミングの初歩
9/25	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基礎と利用方法
調整中	Pythonチュートリアル（中級編）	Pythonプログラミングの応用編

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

利用負担金制度

一般利用(学術利用)

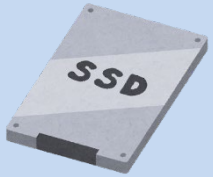
共有利用	
10万円+税	1,000 ポイント
50万円+税	5,250 ポイント
100万円+税	11,000 ポイント
300万円+税	34,500 ポイント
500万円+税	60,000 ポイント

占有利用	
1,150,000 円+税	汎用CPU 1ノード/年

＋

産業利用
成果公開型

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
初期容量なし
5,000円/TB で追加可能

産業利用
成果非公開型

金額 × 5

詳細は　https://www.hpc.cmc.osaka-u.ac.jp/service/cost/

スパコンの提供方法

共有利用

「ノード時間」or
「SQUIDポイント」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

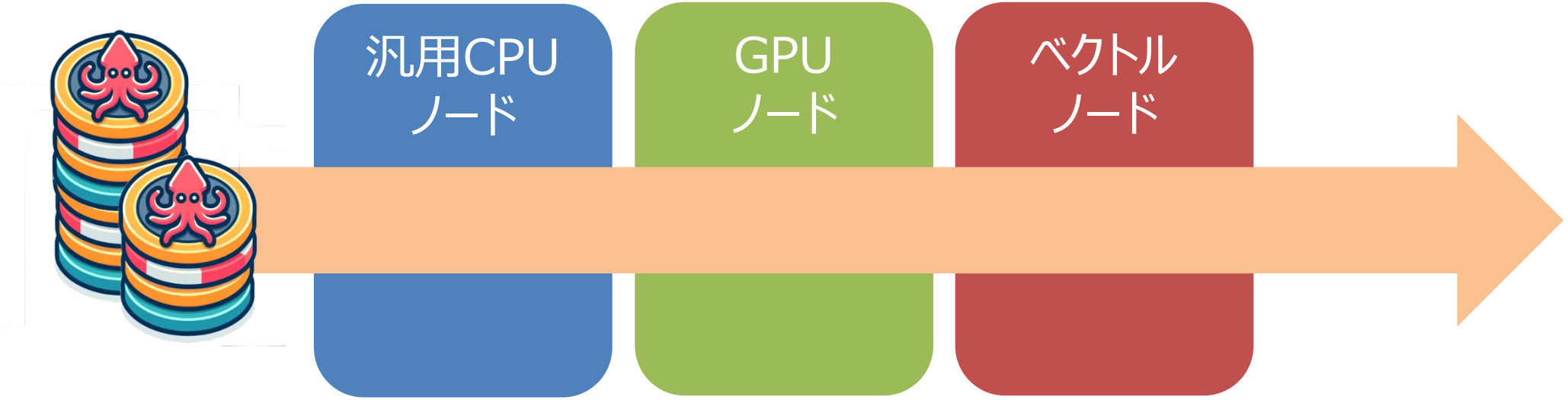
他のグループとノードを共有しない

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

「SQUID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

- 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



「SQUIDポイント」とは

SQUID 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

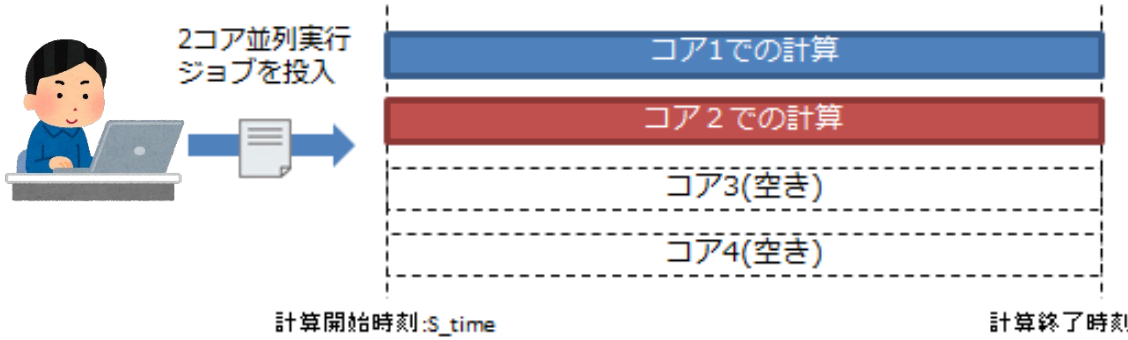
「ノード時間」とは

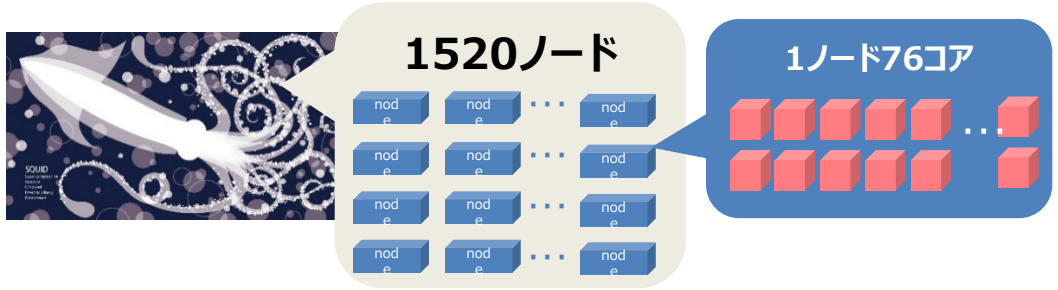
ノード時間 ＝
　計算に使用するノード数 × 計算時間(単位:時間)

（例）

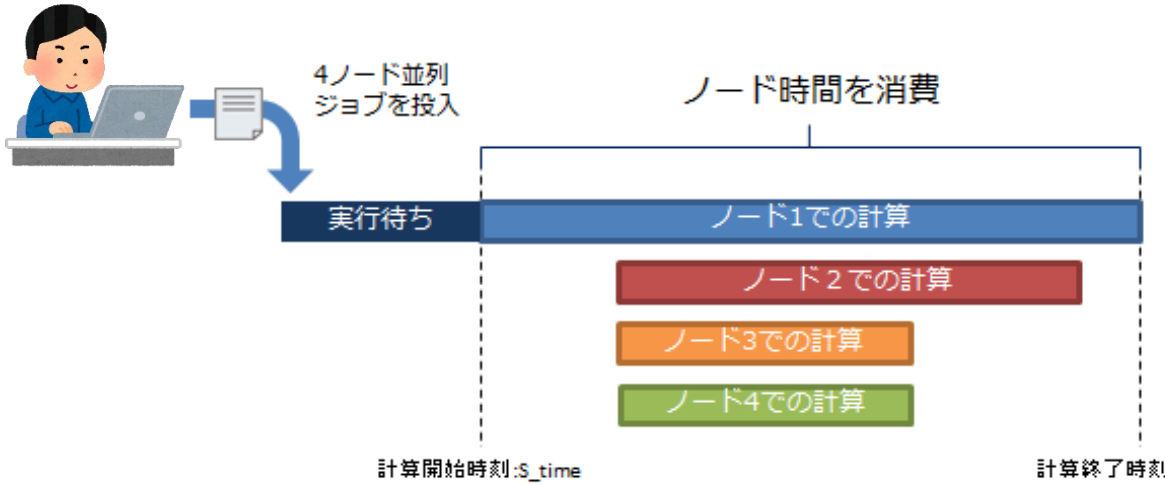
1ノードで3時間の計算	→	3ノード時間消費
30ノードで5時間の計算	→	150ノード時間消費
100ノードで1時間の計算	→	100ノード時間消費
1ノードで100時間の計算	→	100ノード時間消費

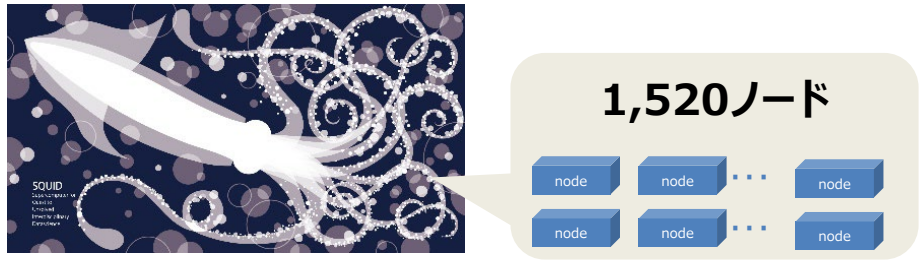
「ノード時間」とは





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

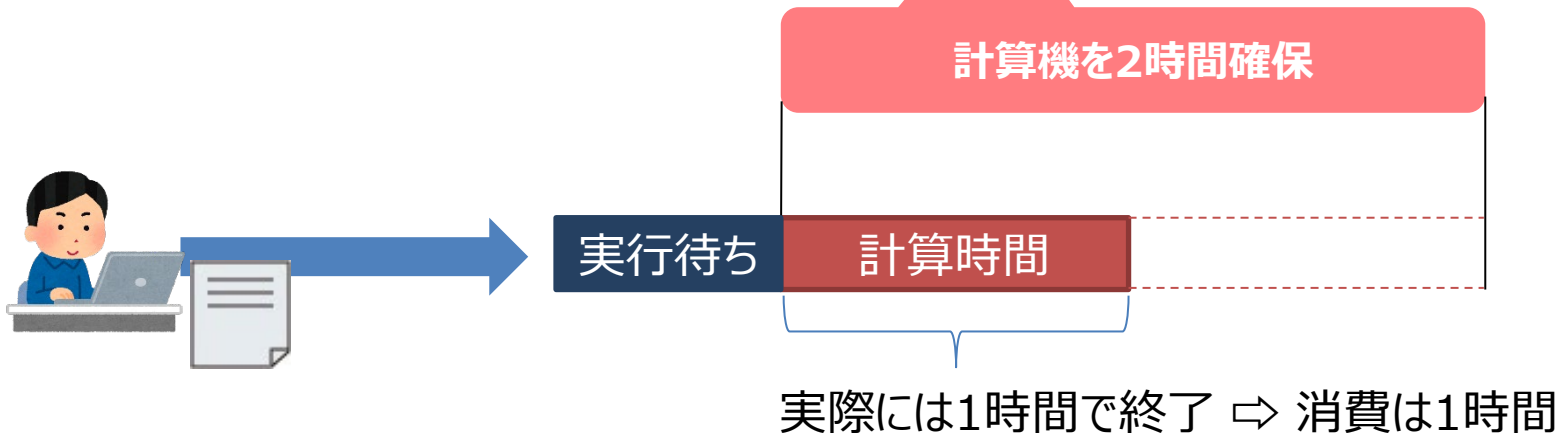




ノード時間は4ノード × (計算終了時間 - 計算開始時間)です

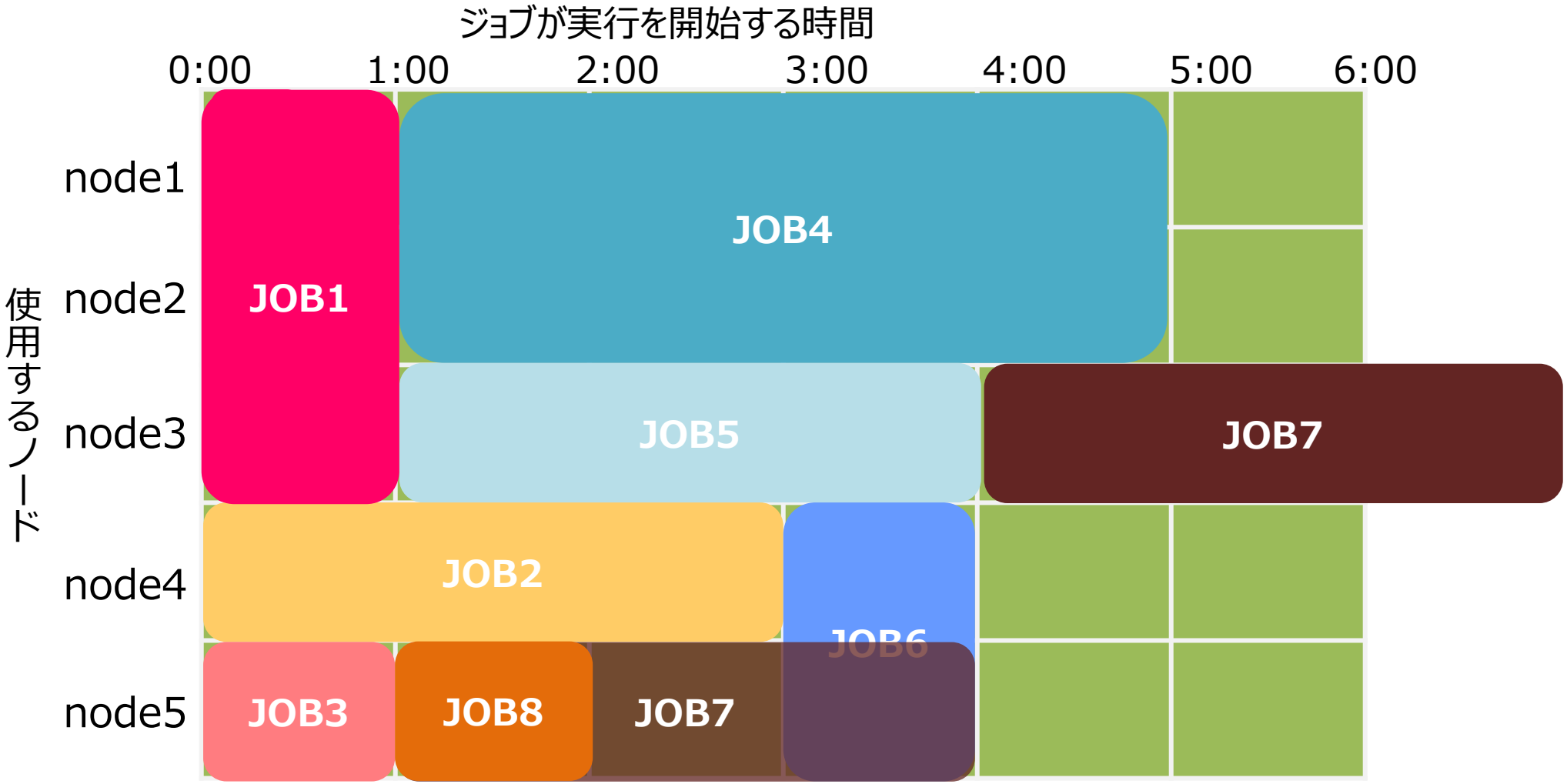
「ノード時間」とは

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



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

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

ノード群	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

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（2024年4月1日～2025年3月31日）				燃料係数
	4-6月	7-9月	10-12月	1-3月	
汎用CPUノード群	1.0	1.0	1.0	1.0	0.85（2024年4月時点）
GPUノード群	1.0	1.0	1.0	1.0	
ベクトルノード群	1.0	1.0	1.0	1.0	

（例）2023年度4月～6月の利用率が低い
→2024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 ポイントの例

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

ノード群	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

季節係数・燃料係数

ノード群	季節係数				燃料係数
	4-6月	7-9月	10-12月	1-3月	
汎用CPUノード群	1.0	1.0	1.0	1.0	0.85 (2024年4月時点)
GPUノード群	1.0	1.0	1.0	1.0	
ベクトルノード群	1.0	1.0	1.0	1.0	

SQUID 汎用CPUノードを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ポイントの目安

10万円コースで利用できるノード時間の目安
(通常優先度で実行した場合)

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

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 5TB

汎用CPU ノード	GPU ノード	ベクトルノード
294 ノード時間	48 ノード時間	78 ノード時間

- 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

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！

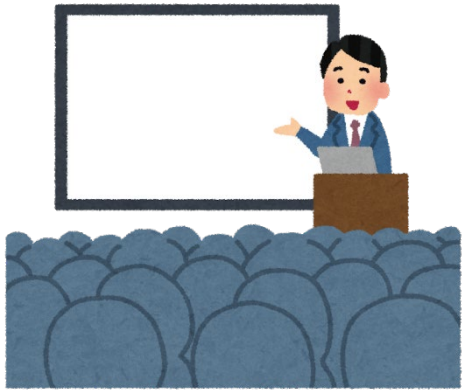
利用は年度単位(4月から翌年3月まで)

- 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

スパコンの使い方のまとめ

- 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！