大規模計算機システムの利用方法

初めてのスパコン

2025年10月10日

大阪大学D3センター
大規模計算機システム担当

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

目次

1. 大規模計算機システムの種類

2. システムの利用方法

3. 大規模計算機システムの正式利用について

4. 試用利用／お試し無料利用と申請方法について

5. 技術サポートについて

6. 問い合わせ先

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

1. 大規模計算機システムの種類

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

SQUIDについて

3種類のノードと21PBのストレージで構成された、
ハイブリッド型スーパーコンピュータ。
各ノード間は200Gbpsで通信可能

	汎用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	25.61 TFLOPS
メモリ	256GB	512GB	VH:128GB VE:384GB
ノード数	1520ノード	42ノード(8GPU /ノード)	36ノード (8VE / ノード)

OCTOPUSについて

汎用CPUノードに絞りSQUIDより高性能に特化した

スーパーコンピュータ

各ノード間は200Gbpsで通信可能

	汎用CPUノード
コア数	140
演算性能	16.384 TFLOPS
メモリ	768GB
ノード数	140ノード

2. システムの利用方法

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

利用の流れ

ユーザ端末

フロントエンドノードへの接続

OCTOPUS
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

OCTOPUS
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

octopus.hpc.osaka-u.ac.jp

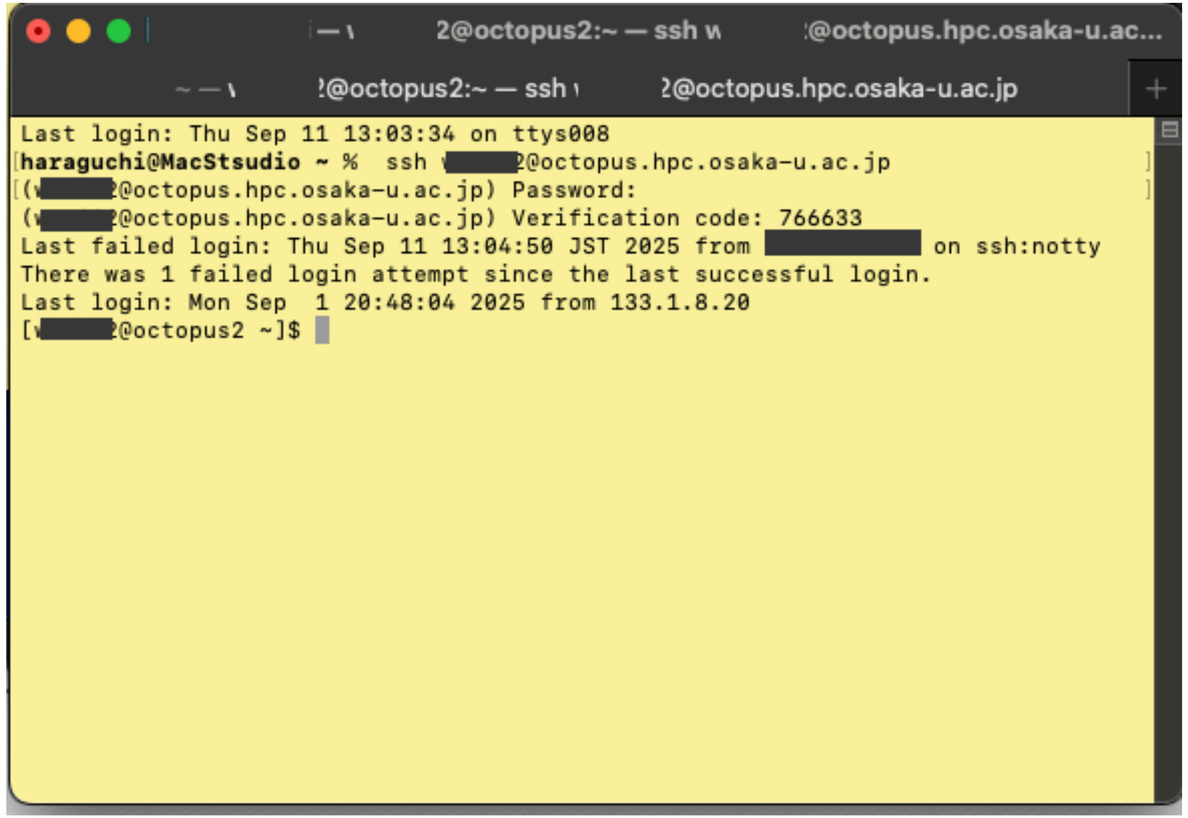
squidhpc.hpc.cmc.osaka-u.ac.jp

接続コマンド例

ssh 利用者番号@octopus.hpc.osaka-u.ac.jp

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

場所を問わずどこからでも接続可能



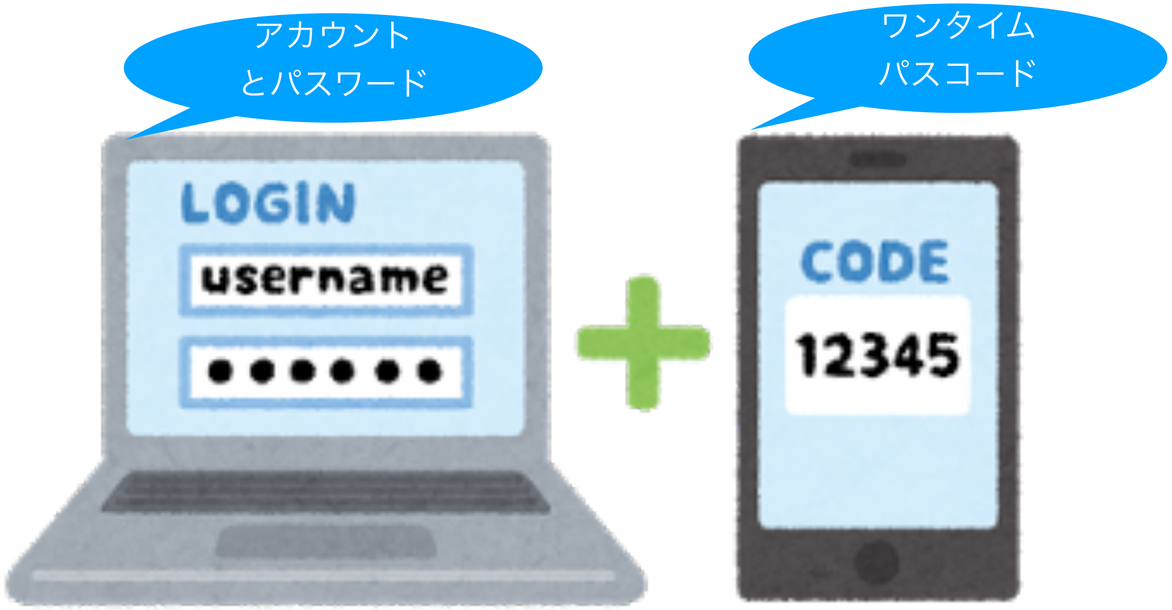
https://www.hpc.cmc.osaka-u.ac.jp/system/manual/octopus2-use/

https://www.hpc.cmc.osaka-u.ac.jp/system/manual/squid-use/

フロントエンドノードへの接続

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

フロントエンドノードへの接続

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

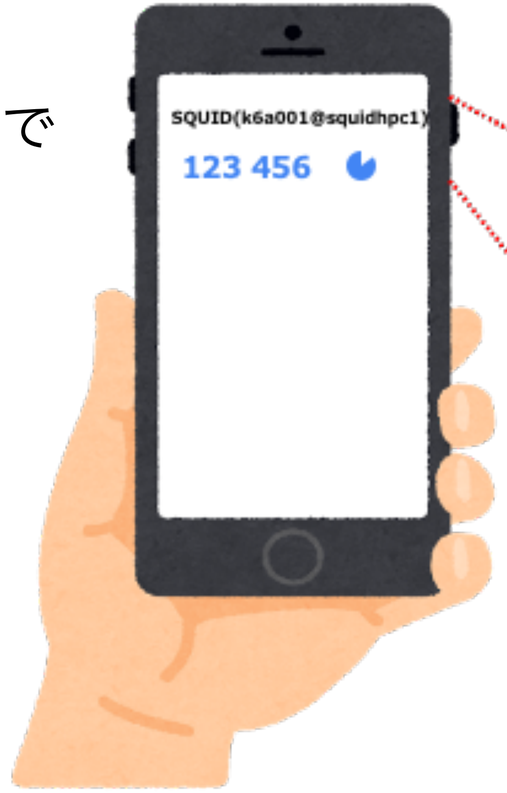
以下いずれかのアプリケーションをインストールしてください

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

基本的に無償で利用可能

フロントエンドノードへの接続

OCTOPUS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



初回登録時のみ

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: XXXXXXXXXXX

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

初回接続時：

$ 利用者番号@octopus.hpc.osaka-u.ac.jp

-> 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登録用アプリで読み込みワンタイムパスコードを取得

-> ログアウトを行う

次回以降の接続：

$ 利用者番号@octopus.hpc.osaka-u.ac.jp

-> パスワード認証を行うと、ワンタイムパスコードの入力が求められます。

-> QRコードを読み込んだアプリを起動し、表示されたパスコードを入力します。

-> 入力が成功すると、フロントエンドノードに接続完了となります。

利用の流れ

ユーザ端末

フロントエンドノードへの接続

OCTOPUS
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

OCTOPUS
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

プログラム準備：環境について

SQUIDでは、多様なソフトウェア・プログラムを実行可能です。

主なソフトウェア：

Gaussian16, GROMACS, OpenFOAM, LAMMPS, TensorFlow, GAMESS, Hϕ, MODYLAS, NTCHem, OpenMX, SALMON, SMASH, PyTorch, Arm Forge, etc.

主なプログラミング言語：

Fortran, C, C++, Python, R, Julia

必要なソフトウェアを、ご自身でインストールすることも可能

プログラム準備：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、

環境の設定が必要となります（Environment modules使を使用）。

	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	BaseEVC	BaseGCC

コマンド例

```
module load BaseCPU
   -> Intelコンパイラの「ifort」が使用可能になる（ifxも使用可能となる）

module load BaseApp
module load gromacs/2021.2
   -> アプリケーション「GROMACS」が使用位可能になる
```

使用可能なモジュールは「module avail」で確認可能

プログラム準備：まとめ

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。

プログラム準備：デモ

1. **サンプルプログラムをコピーする**

   $ cp -p /system/lecture/20250214_AkaiKKR/squid/sample.f90 ~/

2. **Pythonで機械学習をしている方**

   $ module load BaseCPU

3. **FORTRANで書かれたsample.f90 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.f90 もしくは $ ifx sample.f90

利用の流れ

ユーザ端末

フロントエンドノードへの接続

OCTOPUS
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

OCTOPUS
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

1. **インタラクティブ利用**

   コマンド等を通じ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する

   -> 逐次実行のため操作は手軽

こちらを採用

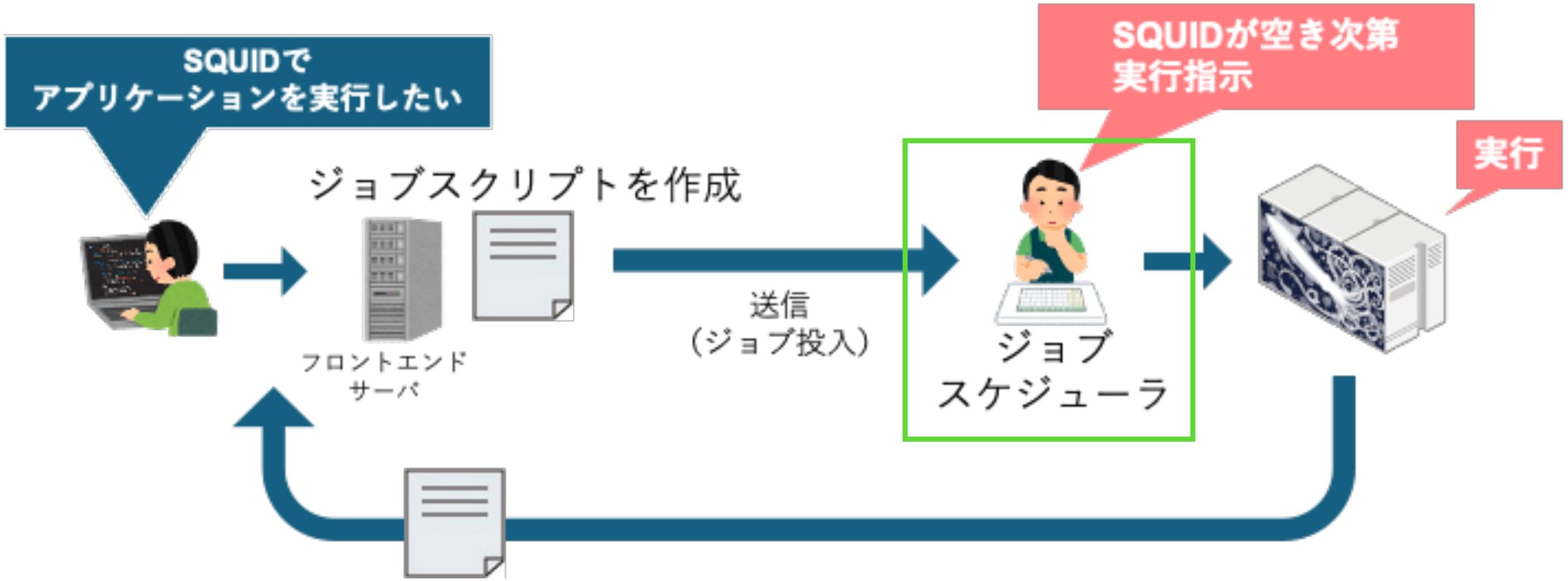
2. **バッチ利用**

   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

-> スクリプトに基づき、計算機が処理を実行する。



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ソフトウェア

主な役割

- 計算機システム各ノードのディスク容量、メモリ容量、性能を把握
- ノード毎の資源使用率を定期的に監視、管理
- ユーザから実行された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

当センターでは**バックフィル型**を採用

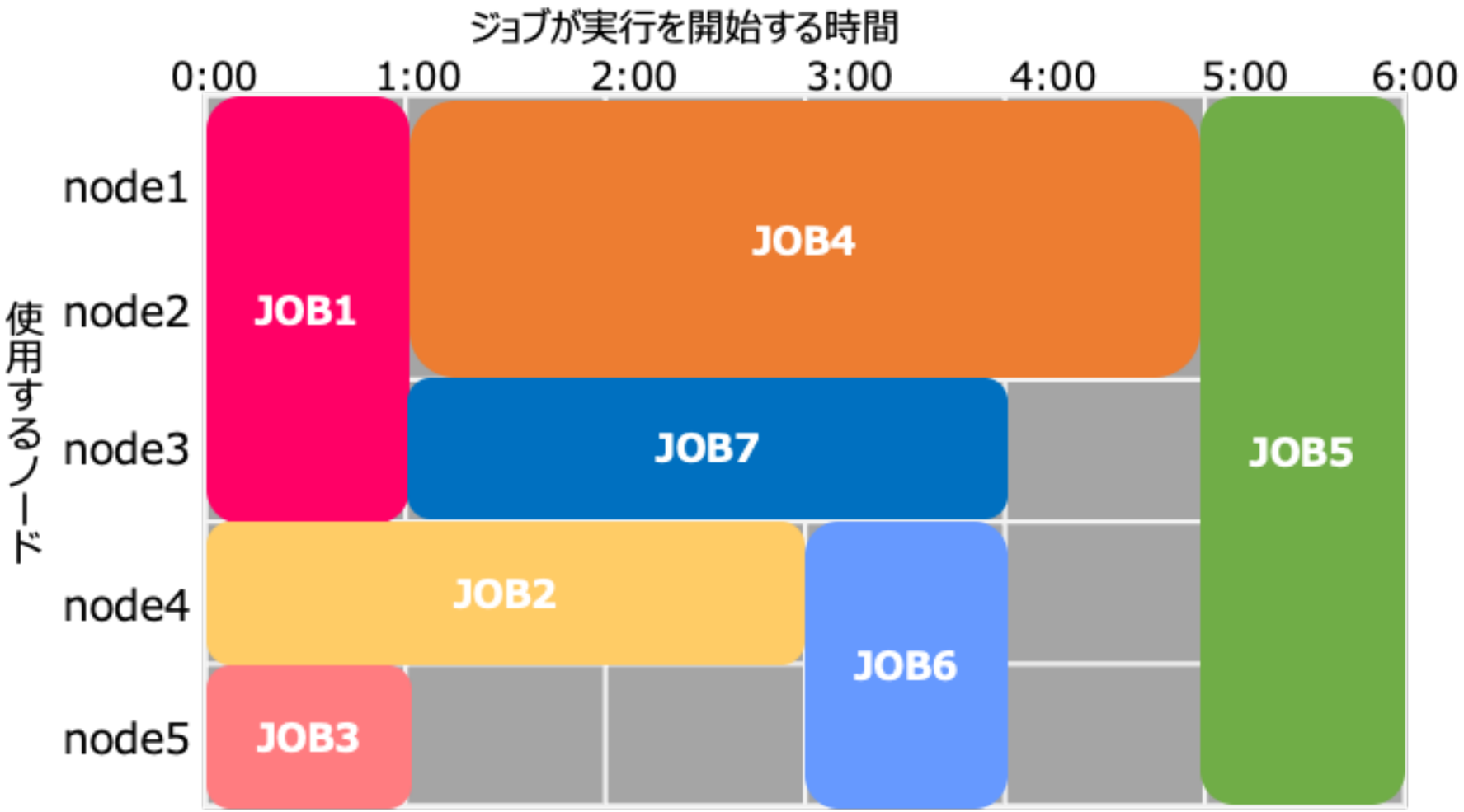
特徴

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

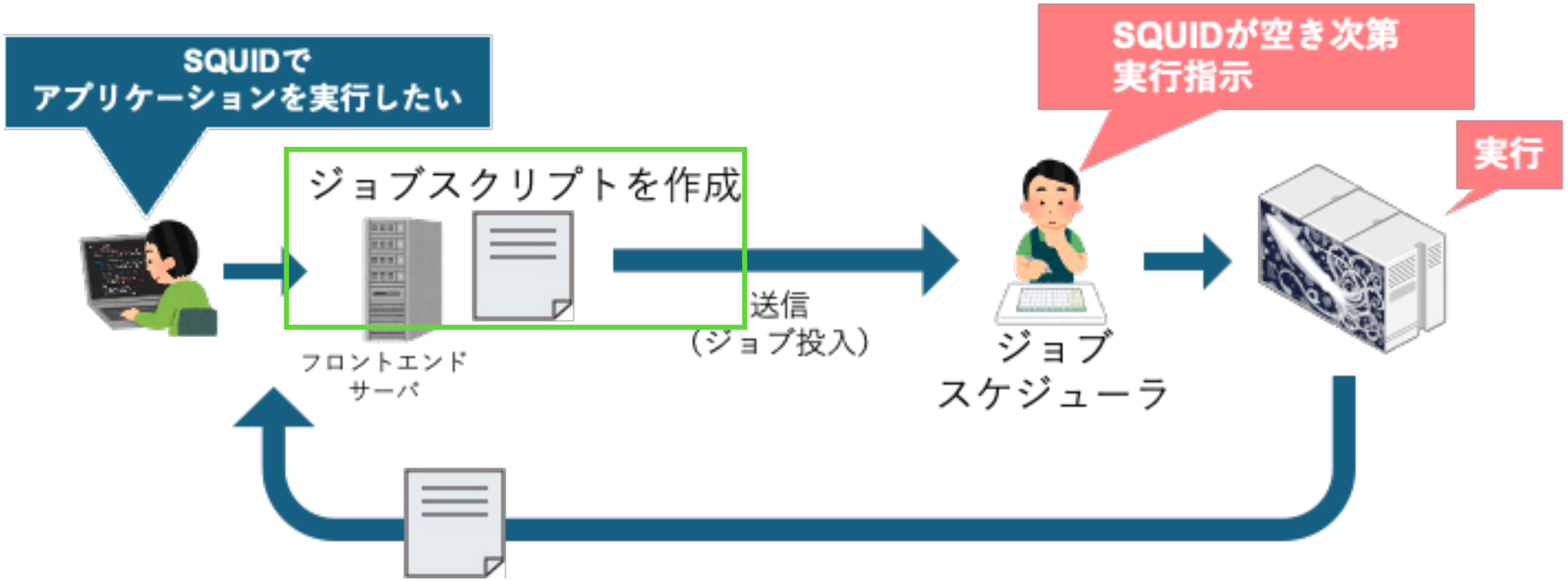
ジョブスケジューラ


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
使用するノード
node1
node2
node3
node4
node5
JOB1
JOB4
JOB7
JOB5
JOB2
JOB6
JOB3

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する。



ジョブスクリプト

```
#!/bin/bash
#PBS -q SQUID
#PBS --group=[グループ名]
#PBS -l elapstim_req=1:00:00
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

計算機のリソースや環境設定を行う。

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

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、計算に使用する計算機やリソースを指定する
#PBS --group	所属するグループ名を指定する(idコマンドで検索可能)
#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 -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)
#PBS -b	使用するノード数

必須パラメータとなります

```
#!/bin/bash

#PBS -q SQUID
#PBS --group=[グループ名]
#PBS -l elapstim_req=1:00:00

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

ジョブスクリプト

```
#!/bin/bash
#PBS -q SQUID
#PBS --group=[グループ名]
#PBS -l elapstim_req=1:00:00
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

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 --group=[グループ名]
#PBS -l elapstim_req=1:00:00
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

利用するノード軍に応じて環境設定が必要となる。

-> 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	BaseGPU	BaseVEC

ジョブスクリプト

```
#!/bin/bash
#PBS -q SQUID
#PBS --group=[グループ名]
#PBS -l elapstim_req=1:00:00
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

ファイルやディレクトリの実行・操作を記述

-> シェルスクリプトを書く要領で記述できる。

- 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 を実行してください。

- $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（まとめ）

```
#!/bin/bash
#PBS -q SQUID
#PBS --group=[グループ名]
#PBS -l elapstim_req=1:00:00
module load BaseCPU
cd $PBS_O_WORKDIR
./a.out
```

- #PBS -q SQUID：ジョブクラスの指定
- #PBS -l elapstim_req=1:00:00：リソースの指定
- module load BaseCPU：環境設定
- cd $PBS_O_WORKDIR：ジョブ投入時のディレクトリへ移動
- ./a.out：プログラム「a.out」の実行

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 –p /system/lecture/nyumon/jobscript.sh ~/

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

   -> $ vi jobscript.sh

※グループ名は kousyuXXX です（XXXは利用者番号のシモ三桁）

例：利用者番号k6b001 -> グループ名：kousyu001

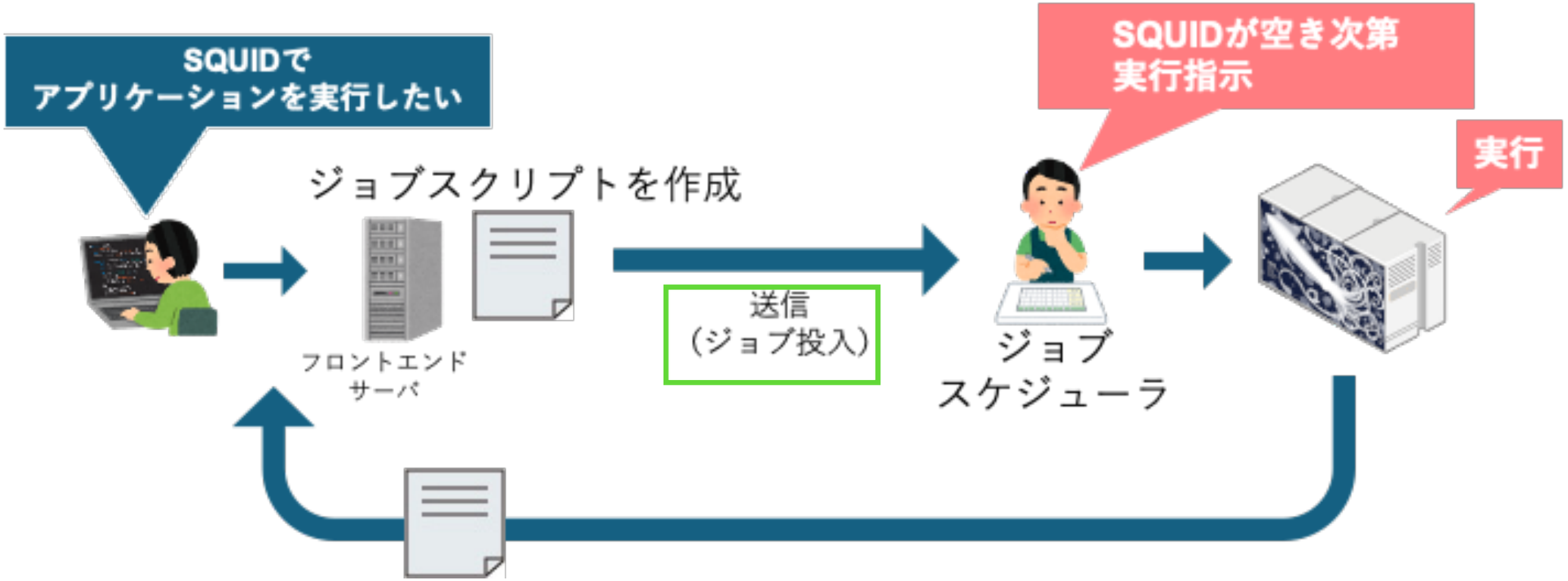
【参考】自身のグループ名は **id** コマンドでも確認できます。

```
$ id
uid=18XX(k6b001) gid=22000(ocean) groups=22000(ocean),14465(kousyu001)
```

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する。



ジョブの操作方法

1. ジョブの投入コマンド

   $ qsub [ジョブスクリプトファイル]

   投入に成功すると、

   “Request [RequestID] submitted to que: ジョブクラス名”

   と表示され、ジョブごとにリクエストIDと呼ばれる通し番号が付与されます。

2. ジョブのキャンセルコマンド

   $ qdel [RequestID]
   キャンセルに成功すると、

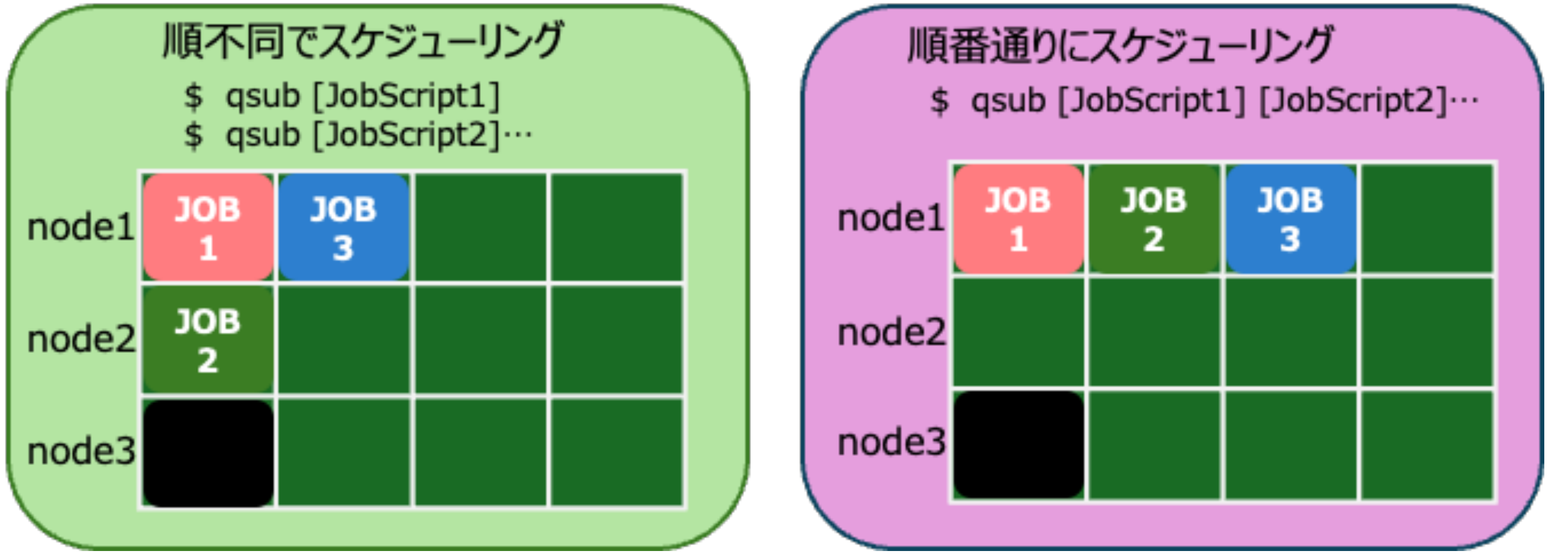
   “Request [RequestID] was deleted”
   と表示されます。

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する

$ 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

ジョブの状態確認コマンド

$ qstat

RequestID	ReqName	UserName	Queue	STT	Memory	CPU	Elapse
12345.sqd	nqs-test	k6a001	SQUID	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

ジョブのスケジューリング状況確認コマンド

$ sstat

RequestID	ReqName	UserName	Queue	Pri	STT	PlannedStartTime
12345.sqd	nqs-test	k6a001	SQUID	-1.5684/ -1.5684	ASG	2024-02-13 13:30:23

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。

実行結果の確認方法

実行結果とエラーは、指定しない限り「標準出力」となる

標準出力：ジョブスクリプト名.**o**リクエストID

標準エラー出力：ジョブスクリプト名.**e**リクエストID

※いずれも自動出力される。

内容確認方法：cat や less コマンド等を使用する。

$ cat jobscript.nqs.o12345

意図通りの出力結果が表示されていればジョブスクリプト実行成功

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. 投入したジョブの状態を確認

   $ cat jobscript.sh.oXXXXX
   $ cat jobscript.sh.eXXXXX

3. 大規模計算機システムの正式利用について

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

利用資格

1. 大学、短期大学、高等専門学校又は大学共同利用機関の教員（非常勤講師を含む。）及びこれに準ずる者
2. 大学院の学生及びこれに準ずる者
3. 学術研究及び学術振興を目的とする国又は地方公共団体が所轄する機関に所属し、専ら研究に従事する者
4. 学術研究及び学術振興を目的とする機関（前号に該当する機関を除く。）で、センターの長（以下「センター長」という。）が認めた機関に所属し、専ら研究に従事する者
5. 科学研究費補助金の交付を受けて学術研究を行う者
6. 第１号、第３号又は第４号の者が所属する機関との共同研究に参画している民間企業等に所属し、専ら研究に従事する者
7. 日本国内に法人格を有する民間企業等に所属する者（前号に該当する者を除く。）で、別に定める審査に基づきセンター長が認めたもの
8. 各号のほか、特にセンター長が適当と認めた者

原則、他の大規模計算機システムと同じ条件となる

https://www.hpc.cmc.osaka-u.ac.jp/service/kitei/

利用負担金制度

一般利用
（学術利用）

OCTOPUS

ディスク容量
HDD：5TB

SQUID

ディスク容量
HDD：5TB

＋

産業利用
成果**公開**

HDD：2,000円/TB

SSD

SSD：5,000円/TB
（SQUIDのみ対応）

産業利用
成果**非公開**

金額　✖　5

計算機の提供方法

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実行可能

-> 「待ち時間」が発生する

共有利用

「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

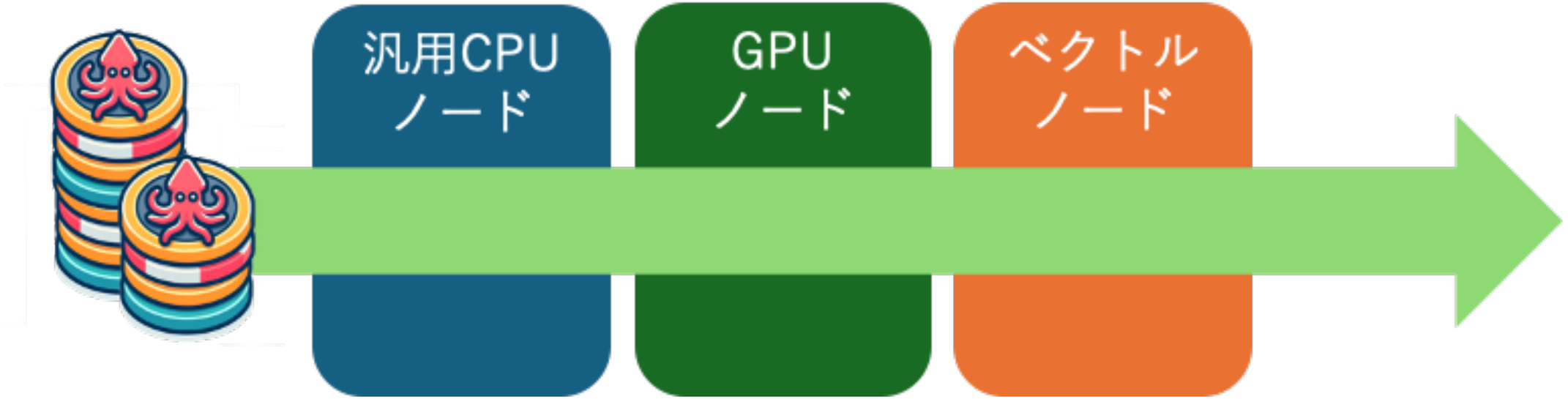
大規模なノード間並列は試し辛い

-> 「待ち時間」が発生しない

ポイント制度

計算ノードの使用時間とノード数に応じてポイントを使用

- 3つのノード軍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



利用負担金制度

- 前払い式
  - コースに応じて右記の
    ポイントが割り当てられます
- 成果非公開型の場合は金額が
  ５倍になります

基本負担額	SQUIDポイント OCTOPUSポイント
10万円	1,000 ポイント
50万円	5,250 ポイント
100万円	11,000 ポイント
300万円	34,500 ポイント
500万円	60,000 ポイント

利用負担金制度

SQUIDポイント／OCTOPUSポイントの消費量は以下の計算式から算出されます

消費ポイント = 使用ノード時間 × 消費係数 × 季節係数 × 燃料係数

https://www.hpc.cmc.osaka-u.ac.jp/system/manual/point/

ノード時間

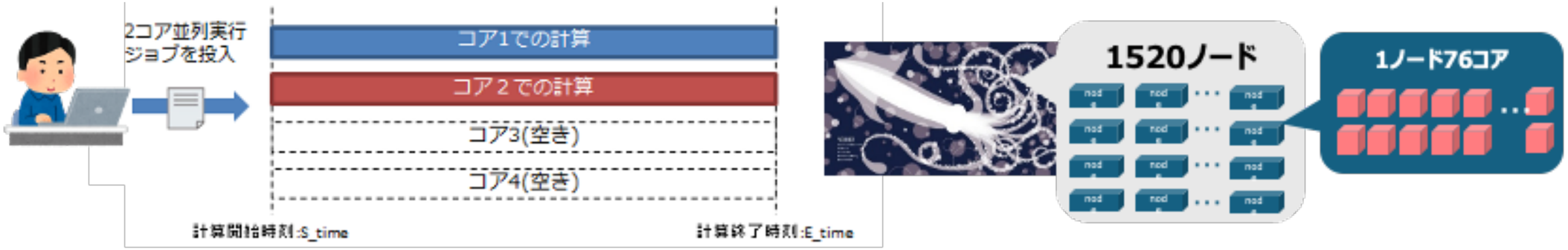
使用するノード数と計算時間をかけたもの

ノード時間 = 計算に使用するノード数 × 計算時間時

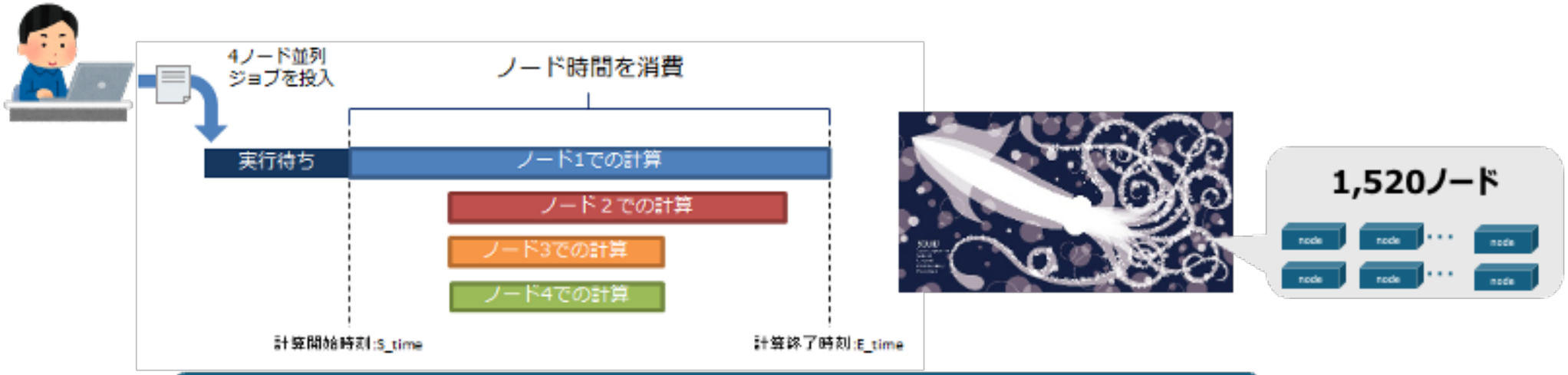
（例）

1ノードで3時間の計算　：　3ノード時間を消費
30ノードで1時間の計算　：150ノード時間を消費
100ノードで1時間の計算　：100ノード時間を消費
1ノードで100時間の計算：100ノード時間を消費

ノード時間



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



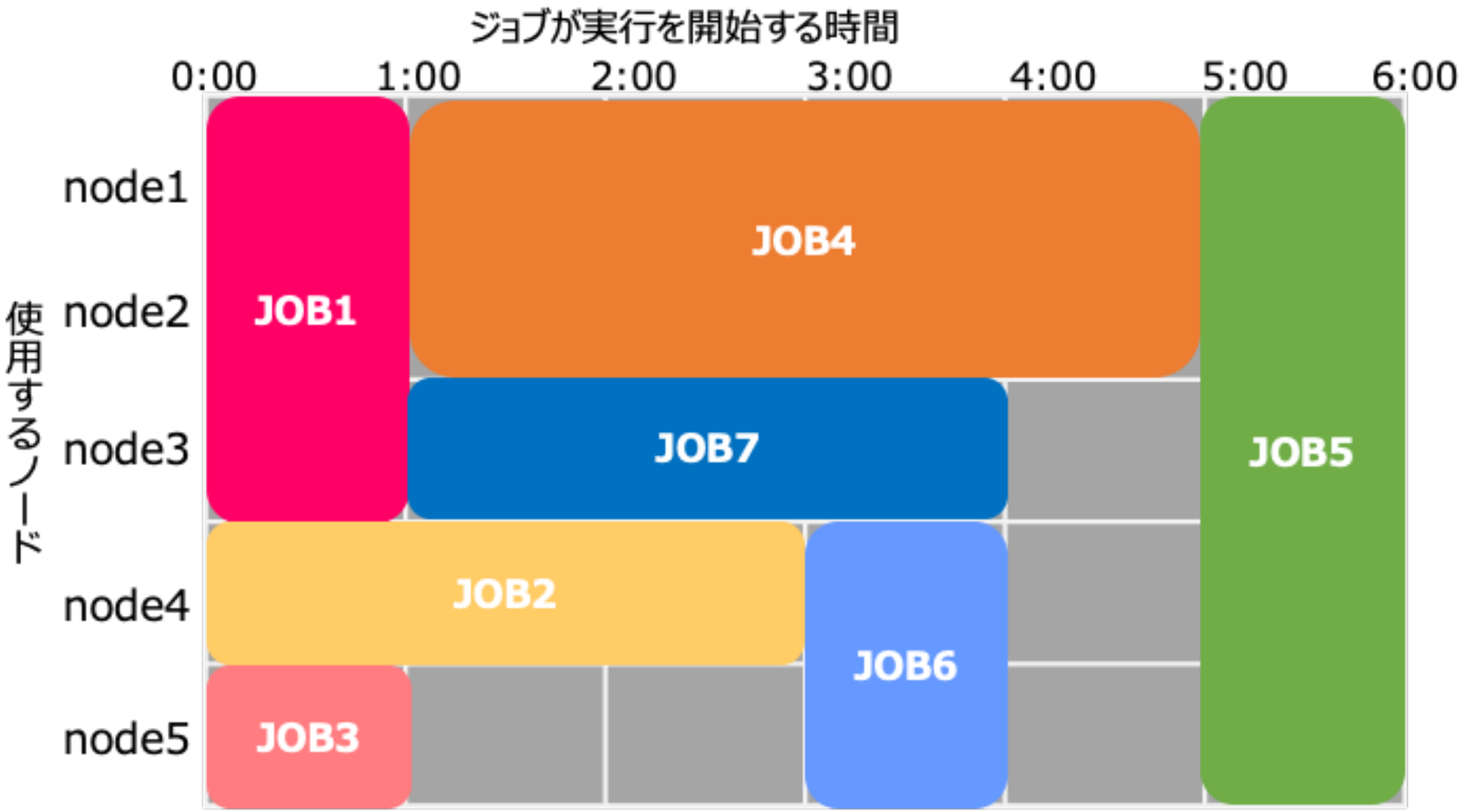
ノード時間は4ノード × (計算終了時間 - 計算開始時間)です

ノード時間

```
#!/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

ジョブスケジューラ


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
使用するノード
node1
node2
node3
node4
node5
JOB1
JOB4
JOB7
JOB5
JOB2
JOB6
JOB3

消費係数

消費ポイント = 使用ノード時間 × 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、
ポイントの消費量は異なる

「季節係数」と「消費係数」について

消費ポイント = 使用ノード時間 × 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（2025年4月1日～2026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）2025年4月～6月の利用率が低い
-> 2026年度4～6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

ポイントの例

消費ポイント = 使用ノード時間 × 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**

ポイントの目安（SQUID）

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**

4. 試用利用／お試し無料利用と申請方法について

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

試用制度について

- 3ヶ月間**無料**で以下の資源をご利用いただけます

SQUID **75 SQUIDポイント** + ストレージ 5 TB

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

- 計算環境や技術サポートは有償利用と同等に使用可能

- 優勝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付中です！
- 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

お試し無料利用について

- 現在利用の方、検討中の方、いずれも使用可能
- 利用者としては無料でスパコンを自由に使うことができる
- 運用側としては、いろんなバグや運用上の不都合などの声を届けてもらい、本運用に向けて改善を進めていく
- SQUIDとの計算速度との違いや、計算で必要なスペックを調べたり
- OCTOPUSで試してみたい方法は使えるかのチェック、など

提供期間と諸費用について

- 提供期間
  - 開始: 2025年9月〜（概ね一週間程度で登録完了）
  - 終了: 2025年11月中旬 (12月からの本番運用のため、11月末に保守を実施します)
- 利用料金
  - 無料
  - ただし、お試し無料利用制度を利用する申請なく、通常申請した場合は課金対象となるため注意

お試し無料利用と本利用との違いについて

- 基本的な違いはなし
  - 無料で使えるのでポイント消費を（強く）意識せず利用可能
- グループで利用はできない（個人での使用感をお試しください）
- 占有ノードはご用意しておりません

申請方法について

新規利用の方

- 所定のフォームから申し込みいただけます

  https://www.hpc.cmc.osaka-u.ac.jp/octopus2_free_trial/

既存利用の方

- 申請代表者の方から申請いただけますと、所属するグループの方全員利用可能となります。利用者の方は申請代表者へご相談ください（フォームは新規利用と同じ）

公募利用の方

- 新規利用の方と同様となります

本運用の切り替えについて

		OCTOPUS無料お試し利用 (2025年9月〜11月)	OCTOPUS有償利用 (2025年12月〜)
A	**2025年度 (現在) SQUIDをご利用中のグループ (一般利用・産業利用)**	**申請代表者が無料お試し利用にお申し込みください** (※1)。グループの全員が既存アカウントでお試し無料利用可能になります。	- 有償利用に移行する場合: OCTOPUSの資源追加申請を行ってください。 - 有償利用に移行しない場合: ジョブ投入が停止され、**OCTOPUS上のデータは2026年3月末に削除されます。**
B	**2025年度 (現在) SQUIDをご利用されていない方**	無料お試し利用にお申し込みください。新規アカウントを発行します。	- 有償利用に移行する場合: OCTOPUSの新規利用申請を行ってください。 - 有償利用に移行しない場合: ジョブ投入が停止され、**無料お試し利用アカウントは2026年3月末に削除されます。**
C	**2025年度 (現在) SQUIDをご利用中のグループ (HPCI課題・JHPCN課題・公募型利用)**	各利用者ごとに無料お試し利用にお申し込みください (**グループ単位での申込はできません**)。新規アカウントを発行します。	同上

※1 申請代表者以外が個人で無料お試し利用に申し込むことは可能ですが、新規アカウントを発行します。そのため、OCTOPUS・SQUID間でファイルシステムの相互アクセスはできません。また、有償利用に申し込まない限り、お試し無料利用期間終了時にアカウントが削除されます。

https://www.hpc.cmc.osaka-u.ac.jp/to-octopus2/

5. 技術サポートについて

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

利用相談

- メール/電話相談/問い合わせフォーム
  - 随時受付
  - 質問例
    - 申請方法がわからない
    - ログインできない
    - コンパイル/実行エラーの原因がわからない
    - 実行してもすごく遅い
    - 使いたいソフトがある…等
- 対面利用相談
  - 月4回程度、事前予約制
  - オンライン形式で実施
  - D3センターの教職員やスパコン開発元のエンジニアに直接相談できる

講習会（定期開催）

講習会名	時間	内容
スパコンに通じる並列プログラミングの基礎	1 時間半	初心者向け・座学のみ （無料アカウント無し）
スパコン利用入門	1 時間	初心者向け
SQUID 高速化技法の基礎	3 時間	ノード内の高速化 SQUIDユーザ向け
並列プログラミング入門（OpenMP）	1 時間	ノード内の並列化
並列プログラミング入門（MPI）	3 時間半	ノード間の並列化

毎年2回同じ内容を開催

より高度な利用に向けて

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/

お問い合わせ

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/

スパコンの使い方まとめ

- 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！

6. 問い合わせ先

OCTOPUS
Osaka university Compute
& sTOrage Platform Urging
open Science

大規模計算機システムに関するお問い合わせ

大阪大学情報推進部情報基盤課
研究系システム班（大規模計算機システム担当）
Mail：system@cmc.osaka-u.ac.jp

または

お問い合わせフォーム
https://www.hpc.cmc.osaka-u.ac.jp/support/contact/auto_form/

までお気軽にご連絡ください！