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、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、しばらく利用可能です
- 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

OCTOPUS

2017年12月運用開始　※今年度で提供終了

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	大容量主記憶搭載ノード
コア数	24	24	64	128
演算性能	1.996 TFLOPS	23.196 TFLOPS	2.662 TFLOPS	8.192 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: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

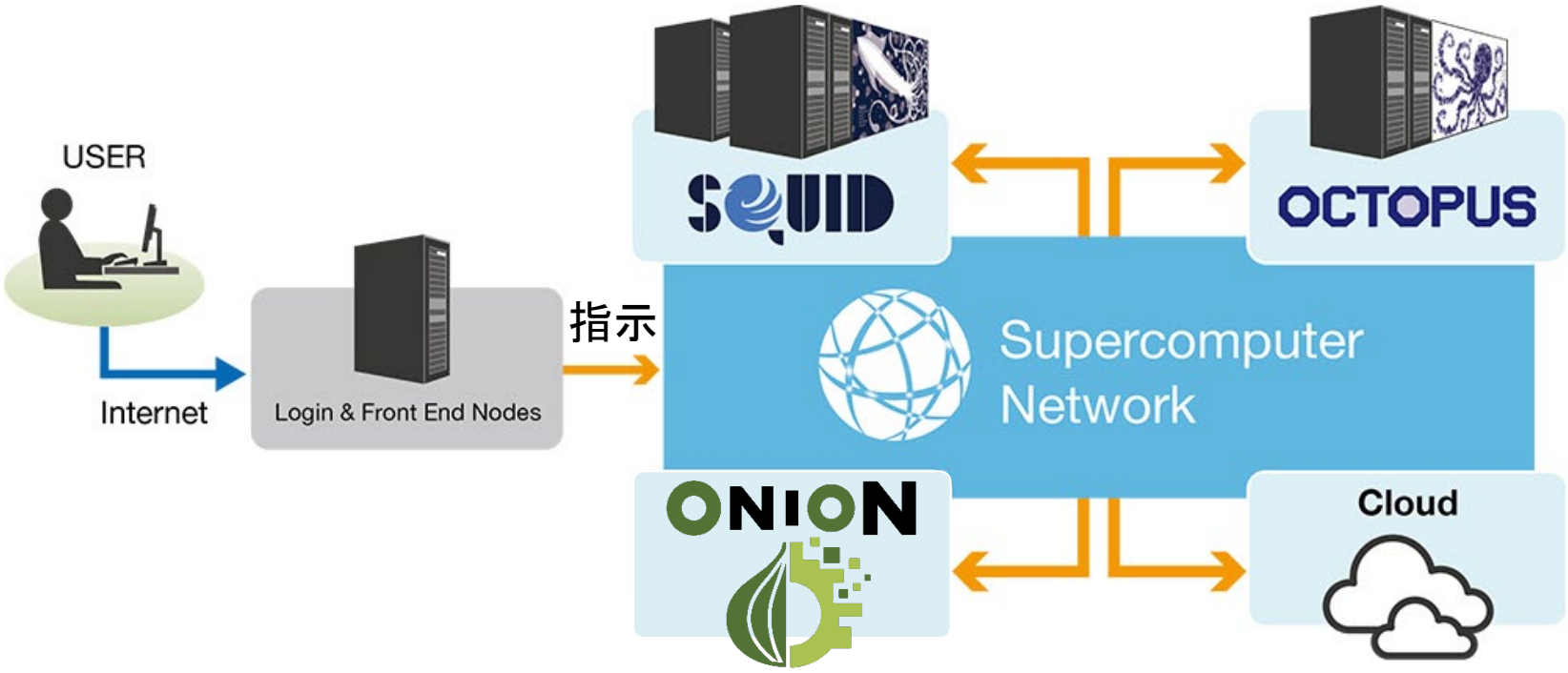
フロントエンドサーバ

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

システムへの接続

フロントエンド端末

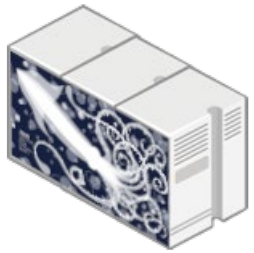
環境設定

プログラム作成

コンパイル

ジョブスクリプト
作成

計算ノード



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

III. 利用を希望する方へ

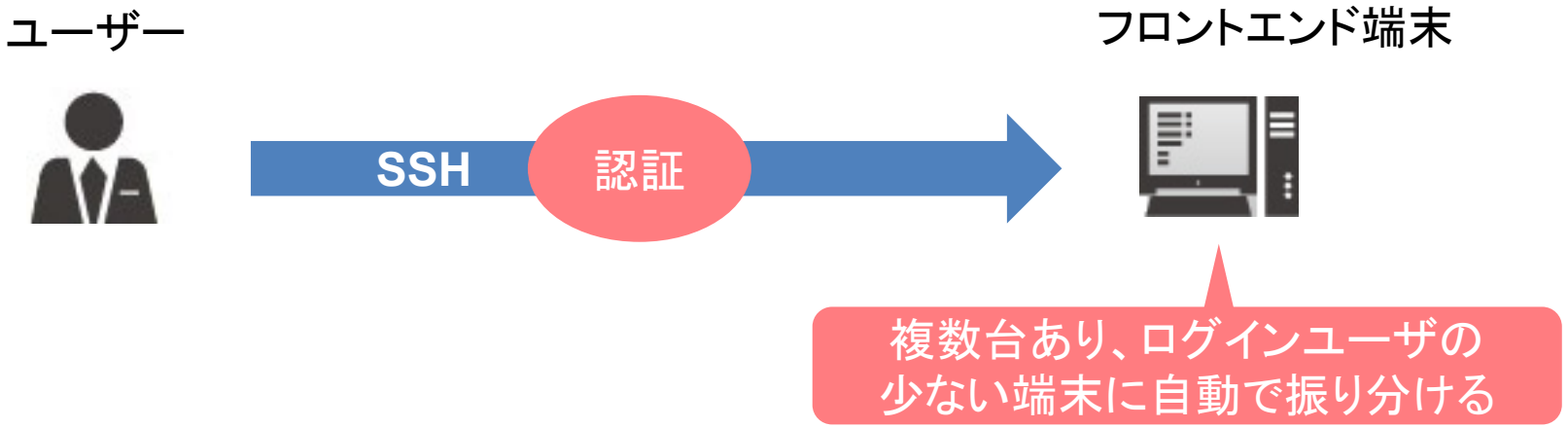
システムへの接続

ログインはSSH（Secure Shell）接続

Win: 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**

SQUID　　：**squidhpc.hpc.cmc.osaka-u.ac.jp**

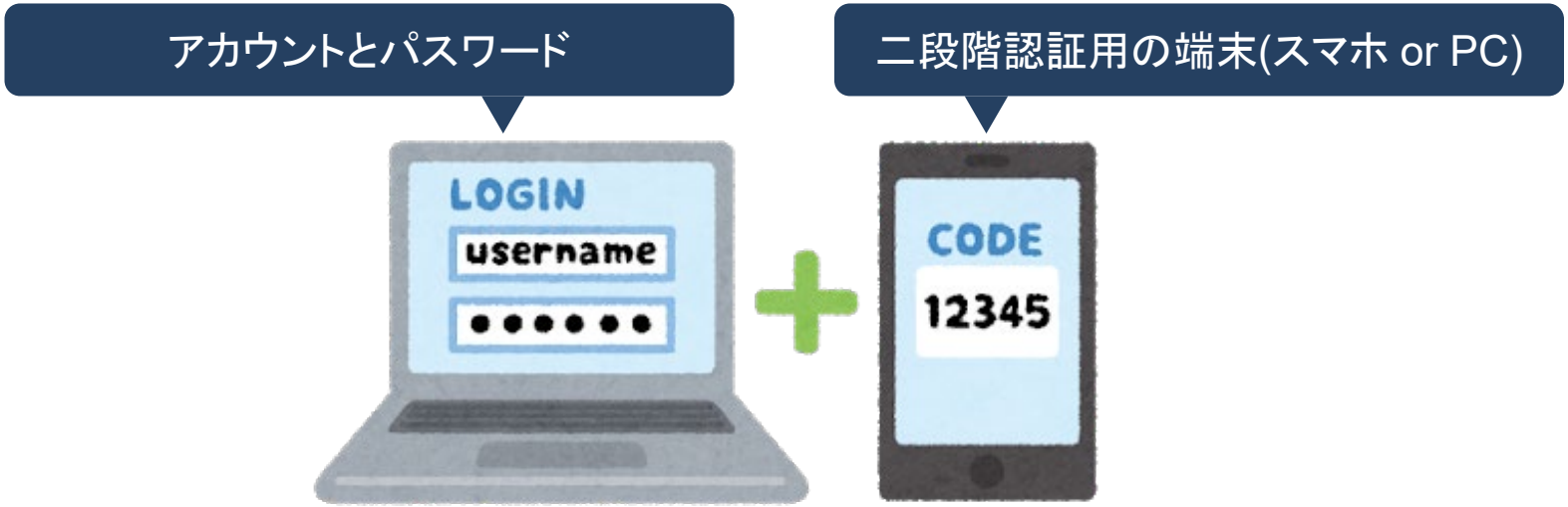


接続コマンド例：ssh アカウント名@squidhpc.hpc.cmc.osaka-u.ac.jp

SQUIDのみ

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



SQUIDのみ

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

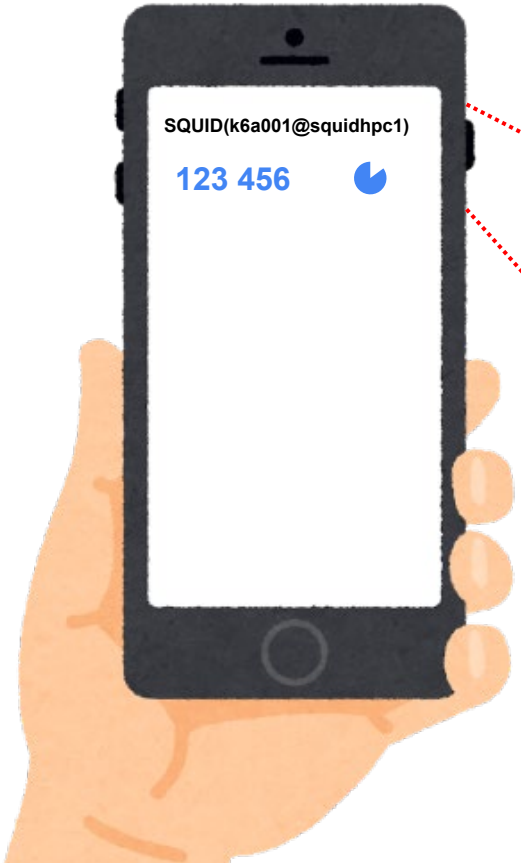
OS	アプリケーション	配布元
Android	Google Authenticator Microsoft Authenticator	Google Play Store
iOS		Apple App Store
Windows	WinAuth	Github
macOS	Step Two	Apple App Store

SQUIDのみ

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



Initiallize 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

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

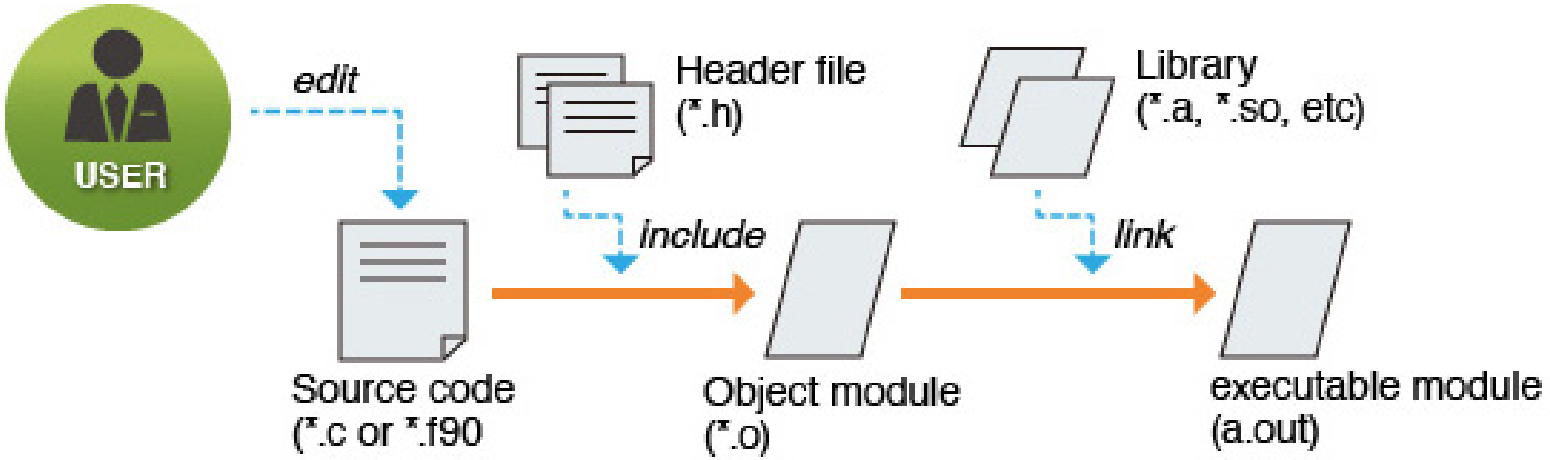
当センターの計算機で使用可能な主な言語

- **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



SQUIDのみ

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（汎用CPUノード向け）	ifort	icc	icpc
NVIDIA HPC SDK（GPUノード向け）	nvfortran	nvc	nvc++
NECコンパイラ（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

デモ1（コンパイル）

1. プログラムの取得（コピー）

   SQUIDの場合

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

   OCTOPUSの場合

   $ cp /octfs/apl/kosyu/nyumon/sample.f ~/

2. 汎用CPUノードの環境設定を読み込み **※OCTOPUSでは不要**

   $ module load BaseCPU/2023

3. sample.f を汎用CPUノード用にコンパイル

   $ 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

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

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

スーパーコンピュータでは使えません。

ただし“会話型風”の機能（インタラクティブ利用）はあり

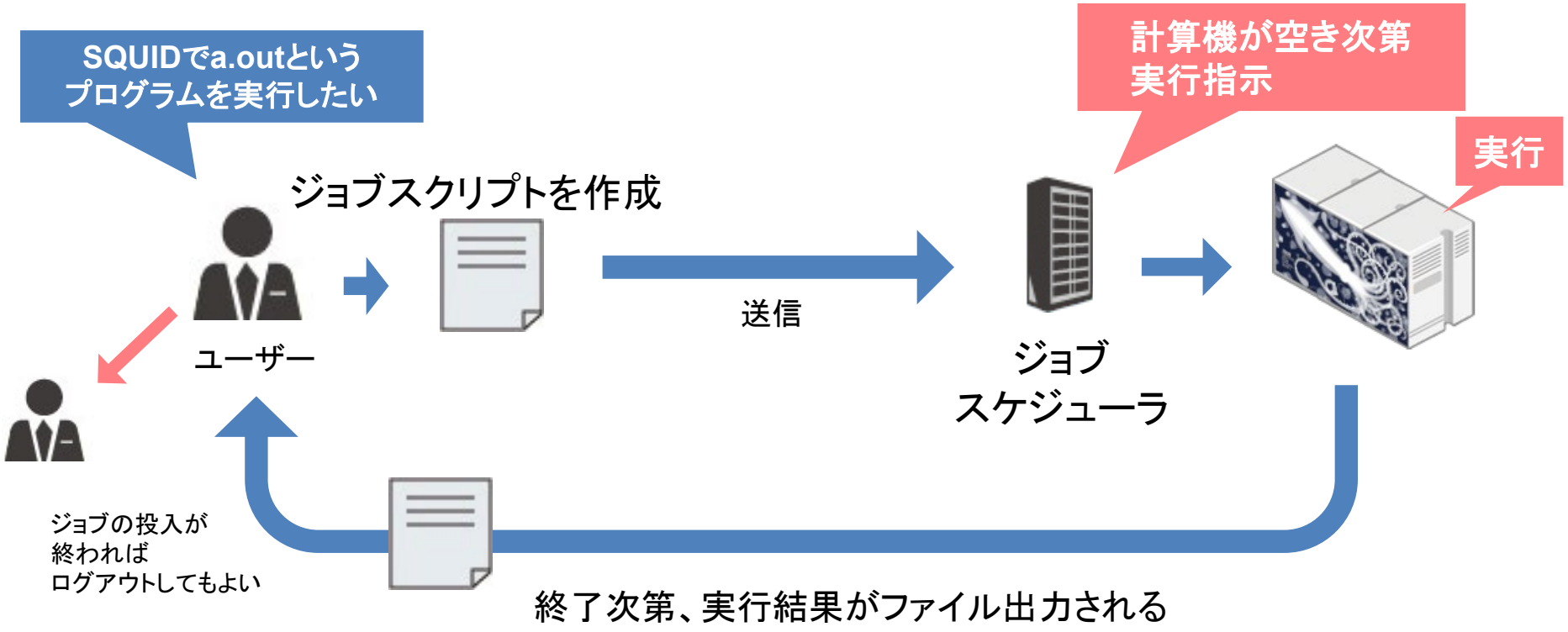
フロントエンド端末での計算実行も禁止

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

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

module load BaseCPU/2023
cd $PBS_O_WORKDIR
./a.out
```

リソース、環境設定の指定

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 ※GPUノードで必須
#PBS --venode	1ノードあたりのVE数 ※ベクトルノードで必須
#PBS --group	グループの指定 ※SQUIDで必須
#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！

ジョブクラス(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	同時利用可能ノード数
OCTOPUS	汎用CPUノード GPUノード	120時間	3,072Core (24Core×128ノード)	128ノード
OCTPHI	Xeon Phiノード	120時間	2,048Core (64Core×32ノード)	32ノード
DBG	汎用CPUノード GPUノード	10分	24Core (24Core×1ノード)	1ノード

ジョブクラス(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
DBG	10分	152Core (76Core×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

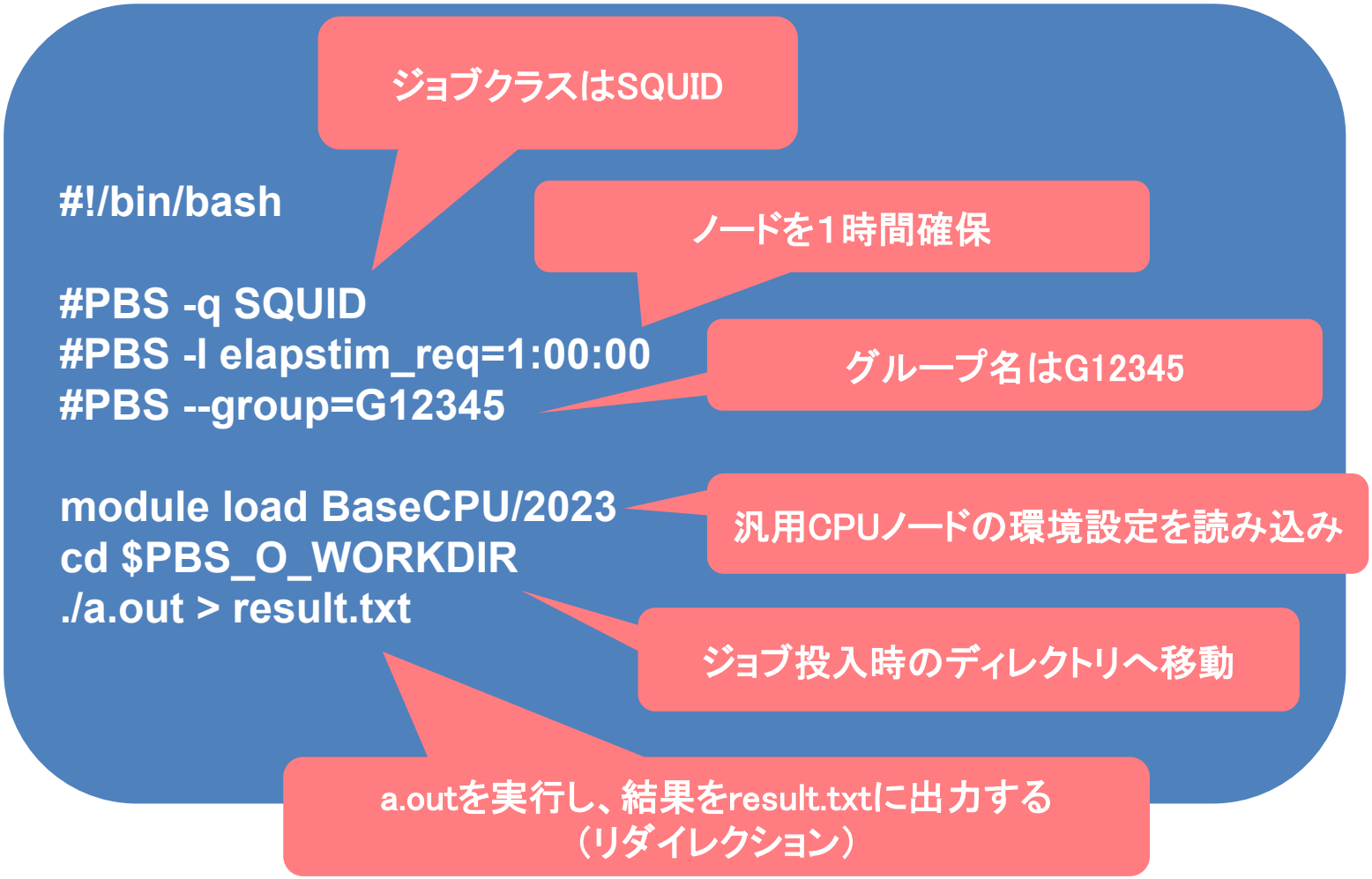
よく使用するNQS 用の環境変数

$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ません！

ジョブスクリプト例（SQUID）



デモ2（ジョブスクリプト）

1. 演習用スクリプトをコピー

   SQUIDの場合

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

   OCTOPUSの場合

   $ cp /octfs/apl/kosyu/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 kosyuXXX です。（XXXは利用者番号の下3桁）

利用者番号：k6a001 ⇨ グループ名：kosyu001

【参考】自身のグループ名は id コマンドでも確認できます。

uid=1805(k6a001) gid=22000(ocean) groups=22000(ocean),14465(kosyu001)

グループ名

本日のプログラム

I. システムのご紹介

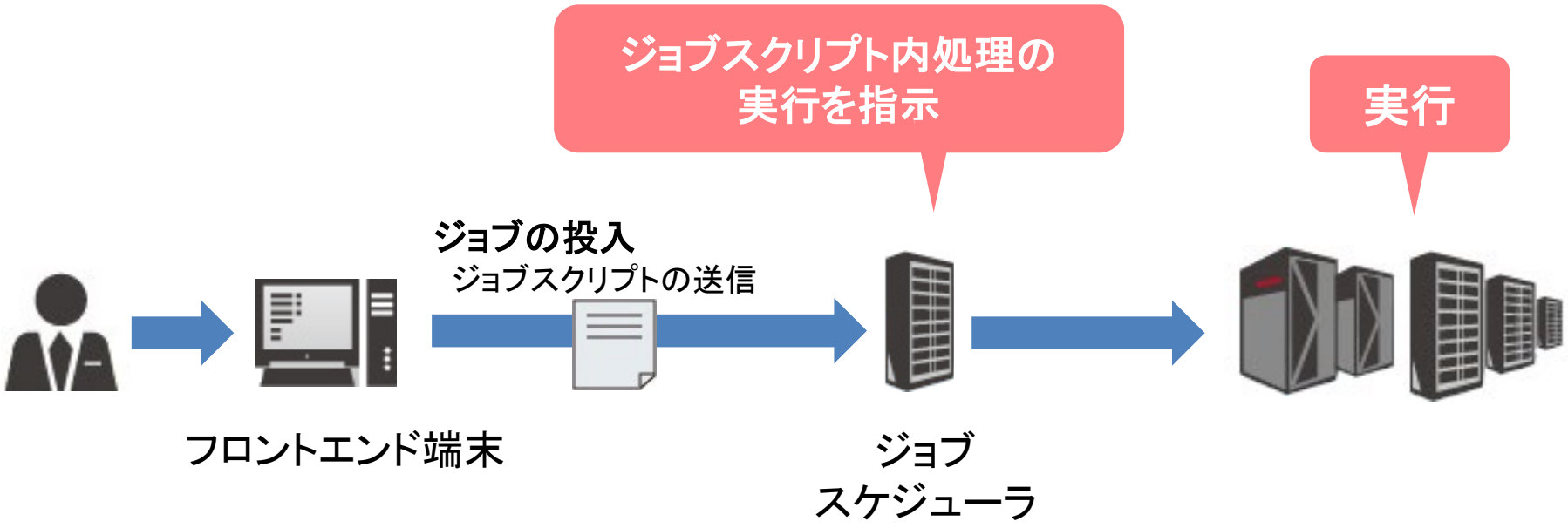
II. 利用方法の解説

- 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

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

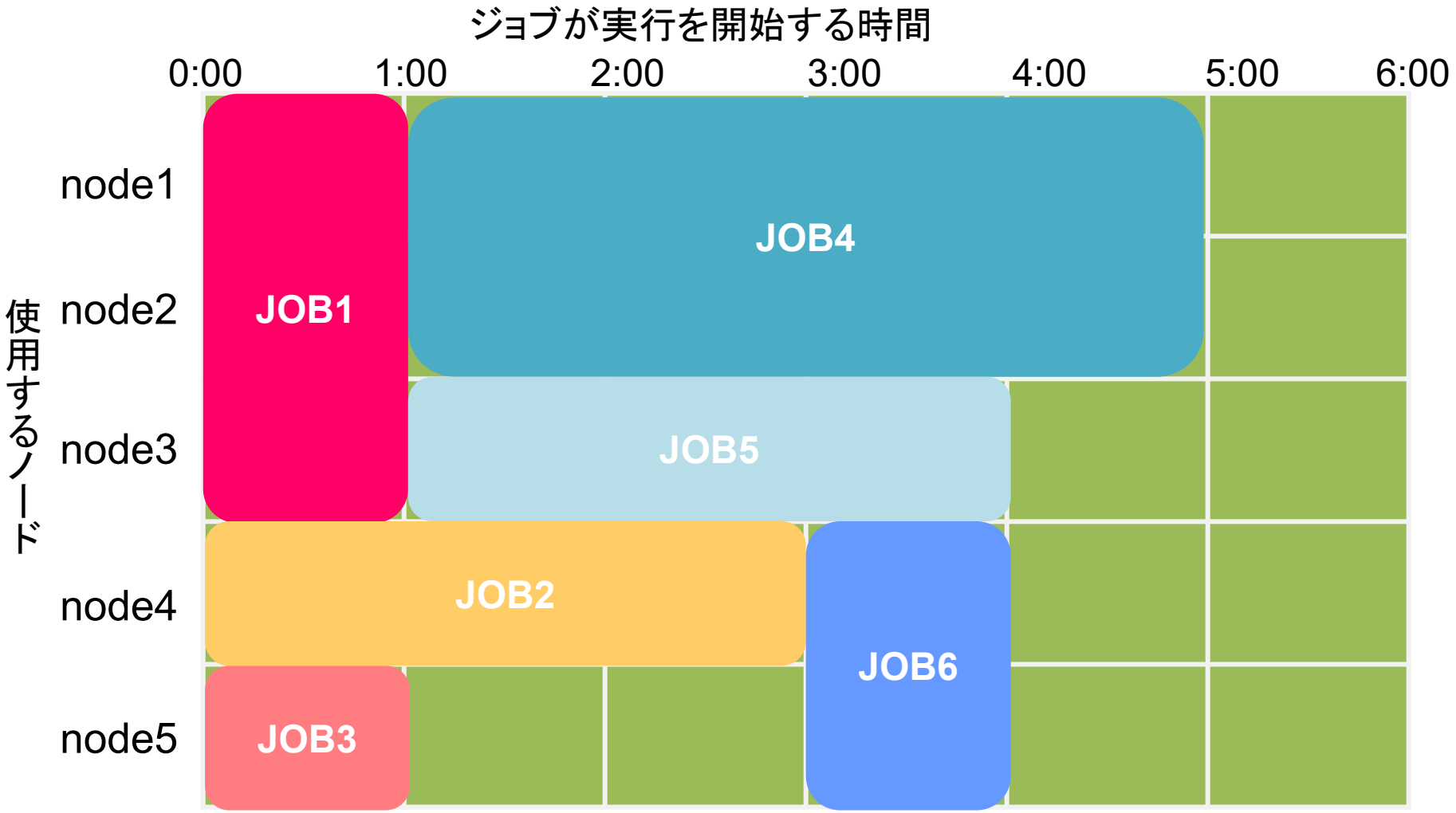
特徴

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



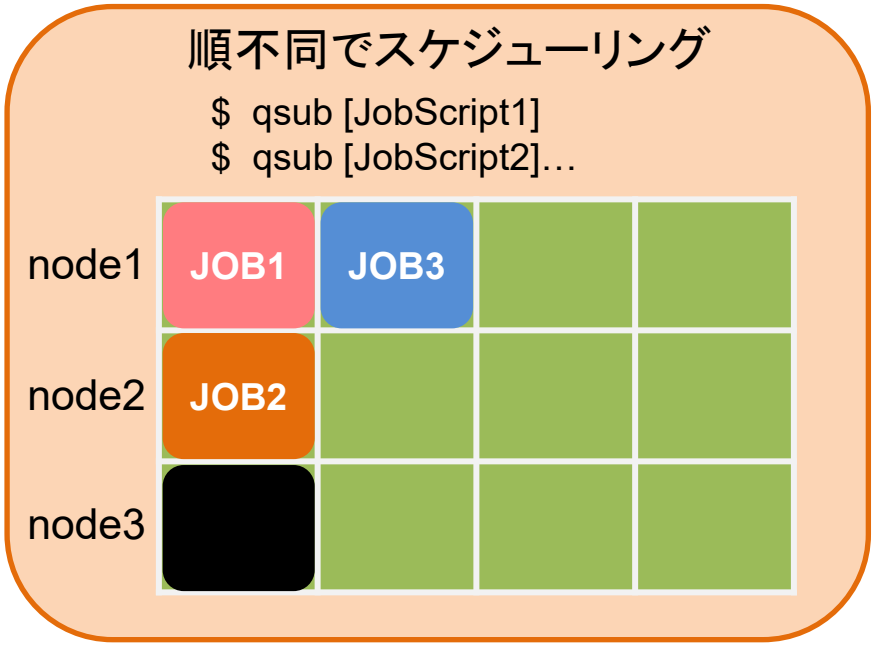
ジョブの投入方法

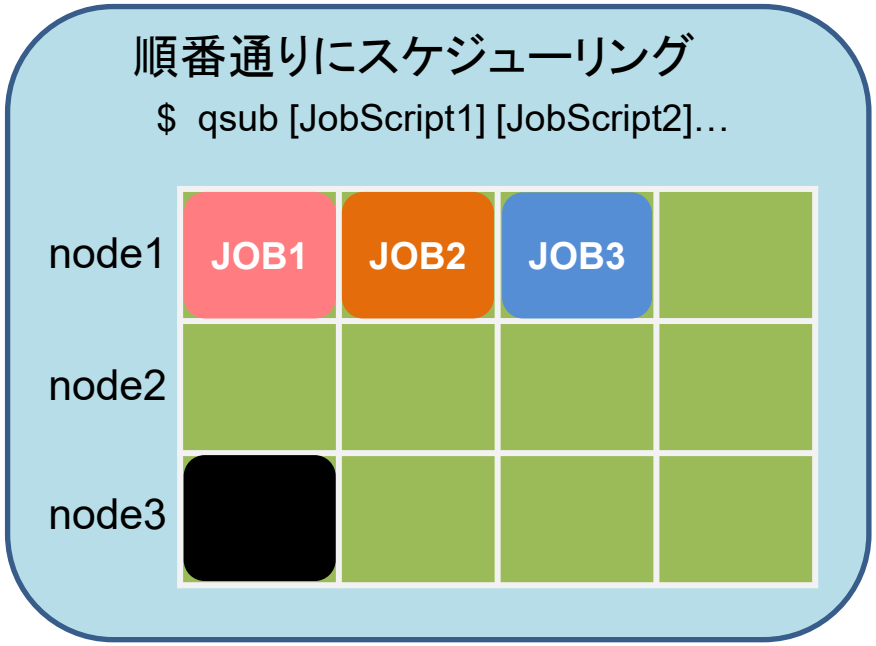
フロントエンド端末からジョブを投入

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





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「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 jobscript.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(>)

デモ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 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

※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

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・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

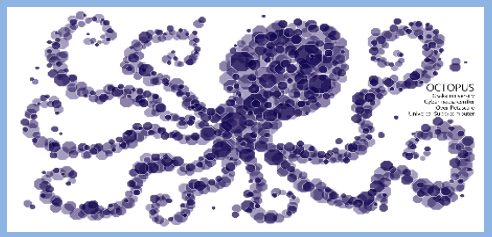
大阪大学　他大学　民間企業

利用負担金が必要になります

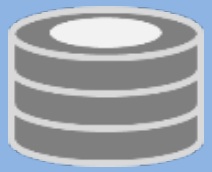
利用負担金制度

一般利用(学術利用)	産業利用 成果公開	産業利用 成果非公開

OCTOPUS



＋



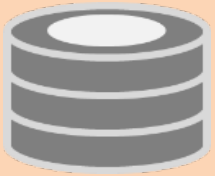
HDDストレージ
初期容量3TB
2,000円/TB で追加可能

ONION

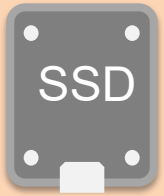
オブジェクトストレージ
12,000円/TB

SQUID

＋



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

金額
× 5

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

利用負担金制度

利用形態は前払い式

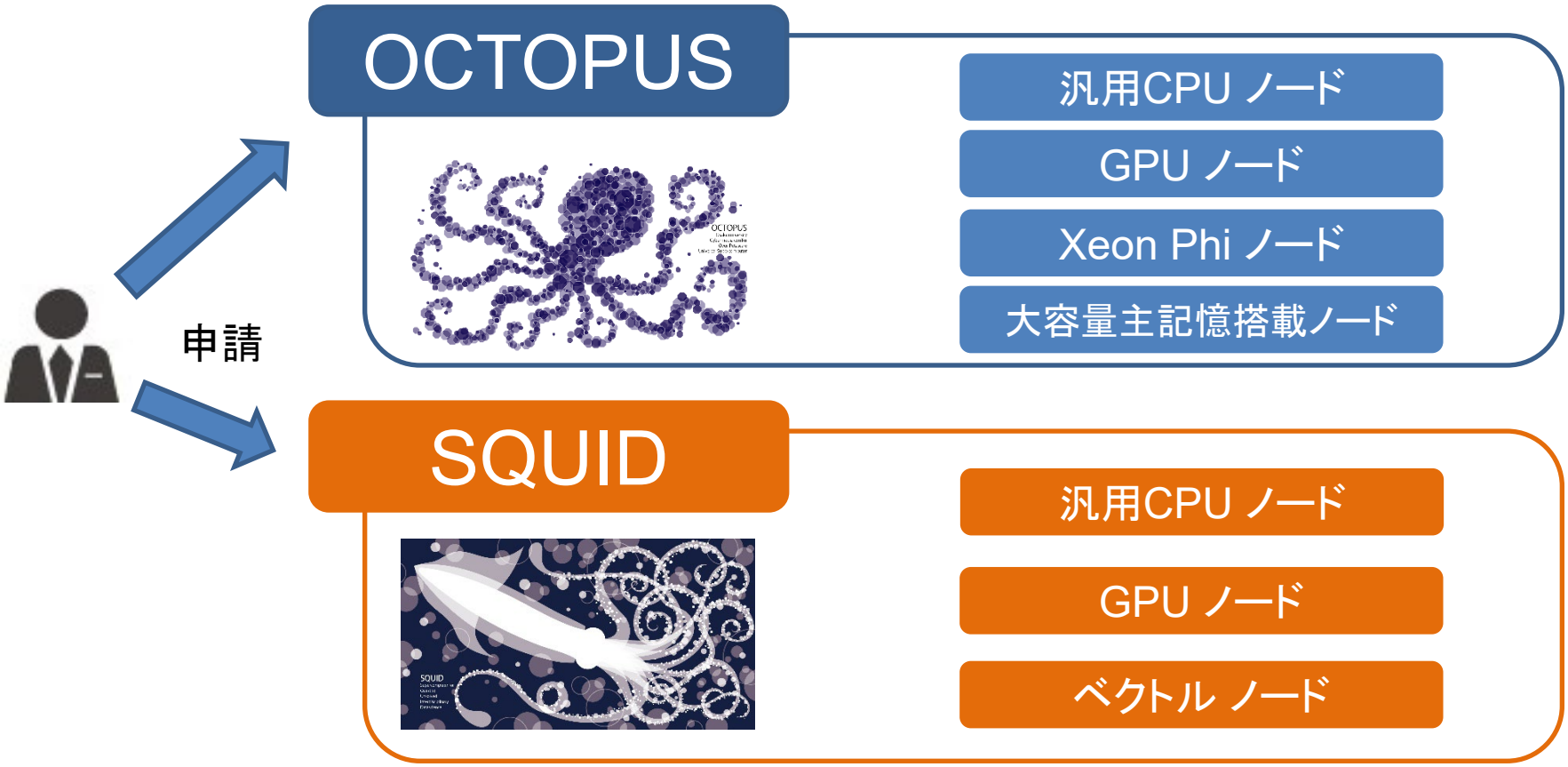
コースに応じて以下のポイントが割り当てられます。

基本負担額	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

※成果非公開型の場合は金額が5倍になります

「OCTOPUSポイント」
「SQUIDポイント」

OCTOPUS、SQUIDへの申請で全てのノードを自由に使用可能とすることを目的に導入された制度です。



「ポイント制度」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

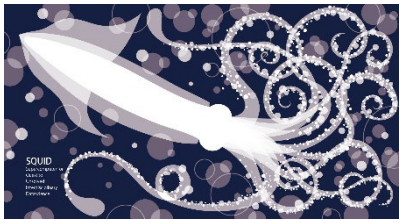
「ノード時間」とは

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

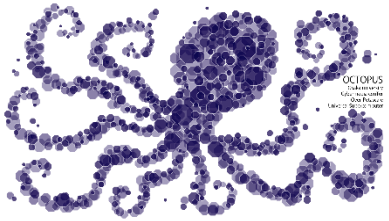
「ノード時間」の補足



1520ノード

node node … node
node node … node
node node … node
node node … node

1ノード76コア



236ノード

node node … node
node node … node

1ノード24コア

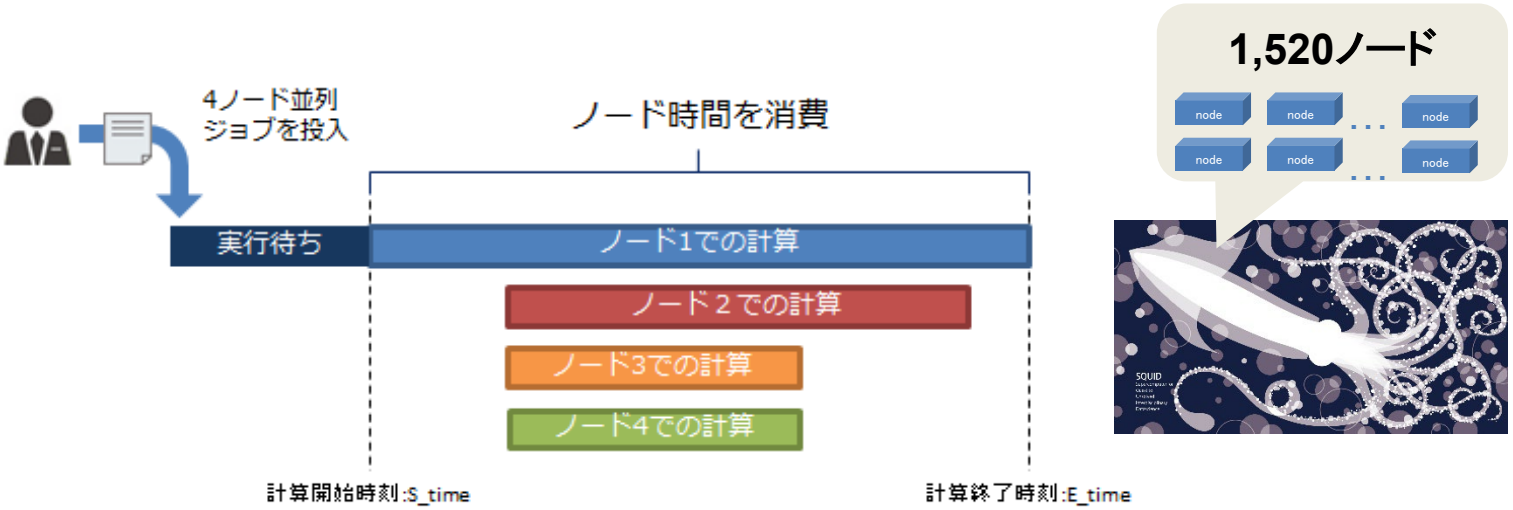
2コア並列実行
ジョブを投入

コア1での計算
コア２での計算
コア3(空き)
コア4(空き)

計算開始時刻:S_time
計算終了時刻:E_time

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

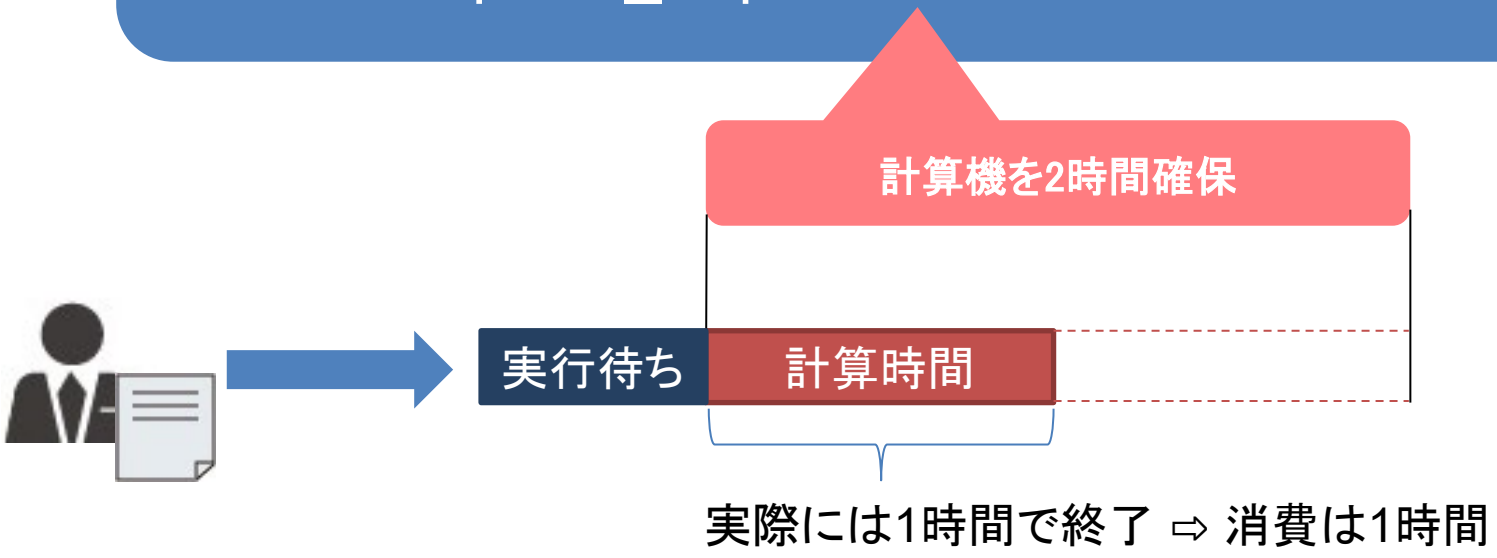
「ノード時間」の補足



ノード時間は4ノード × (計算終了時間 - 計算開始時間)です

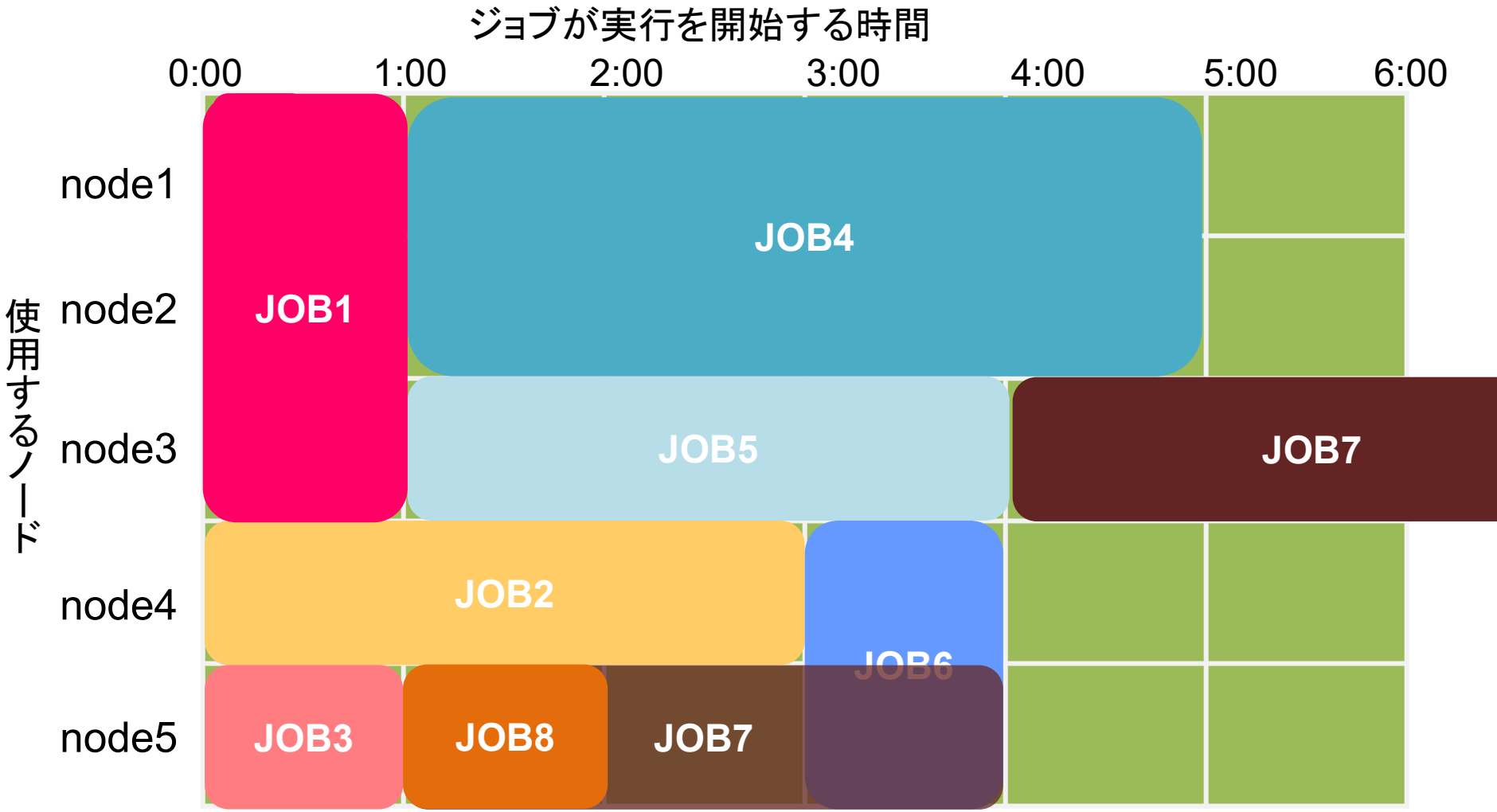
「ノード時間」の補足

```
#!/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

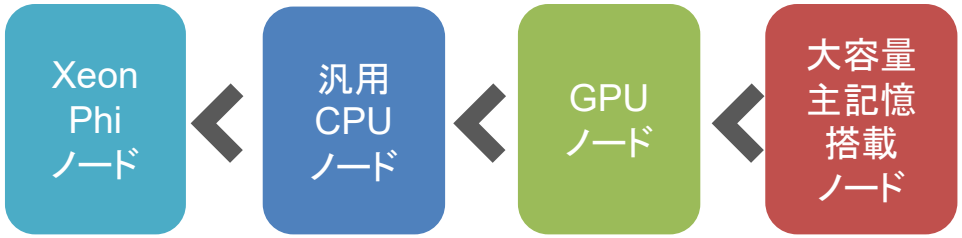
使用したノード時間 × 消費係数 × 季節係数 × 燃料係数

消費係数

各ノード群の消費電力を元に設定

ノード群	消費係数
汎用CPUノード	0.1040
GPUノード	0.4346
Xeon Phiノード	0.0836
大容量主記憶搭載ノード	0.7406

同じノード時間を使用しても、
OCTOPUSポイントの消費量は異なる



「季節係数」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

ノード群	4～6月	7～9月	10～12月	1～3月
CPUノード	1	1	1	1
GPUノード	1	1	1	1
Xeon Phiノード	1	1	1	1
大容量主記憶搭載ノード	1	1	1	1

(例)

2022年度4月～6月の利用率が低かった

2023年度4月～6月の季節係数を低く設定

「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

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	4～6月	7～9月	10～12月	1～3月
CPUノード	1	1	1	1
GPUノード	1	1	1	1
Xeon Phiノード	1	1	1	1
大容量主記憶搭載ノード	1	1	1	1

現在は、一律「1」を設定

→電気料金の変動により、値を変更する場合がある

OCTOPUSポイントの例

消費OCTOPUS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	消費係数
汎用CPUノード	0.1040
GPUノード	0.4346
Xeon Phiノード	0.0836
大容量主記憶搭載ノード	0.7406

季節係数（2023年度）

ノード群	4～6月	7～9月	10～12月	1～3月
汎用CPUノード	1	1	1	1
GPUノード	1	1	1	1
Xeon Phiノード	1	1	1	1
大容量主記憶搭載ノード	1	1	1	1

・汎用CPUノードを10ノード並列実行で3時間使用（季節係数、燃料係数：1）

10 × 3 × 0.1040 × 1 × 1 = 3.120 → **3.120 ポイント消費**

OCTOPUS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

OCTOPUS	ノード群	消費係数	ノード時間
10万円コース 1,000ポイント	汎用CPUノード	0.1040	9,615 ノード時間
	GPUノード	0.4346	2,300 ノード時間
	Xeon Phiノード	0.0836	11,960 ノード時間
	大容量主記憶搭載ノード	0.7406	1,350 ノード時間

※季節係数、燃料係数が 1 の場合

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

SQUID	ノード群	消費係数	ノード時間
10万円コース 1,000ポイント	汎用CPUノード	0.2998	3,335 ノード時間
	GPUノード	1.8348	545 ノード時間
	ベクトルノード	1.1312	884 ノード時間

※季節係数、燃料係数が 1 の場合

まずは試用制度をお試しください

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
311 ノード時間
60 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
41 ノード時間
66 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

利用申請方法

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

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/

研究成果
http://www.hpc.cmc.osaka-u.ac.jp/researchlist/

利用を希望する方へ

本日以降の講習会・セミナー（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 【9月4日(月)】	並列プログラミングの手法や考え方の基礎
Dockerセミナー 【9月6日(水)】	性能測定や基礎的なチューニング手法の説明
今更聞けない数値計算アルゴリズム :常微分方程式編 【9月11日(月)】	常微分方程式に対する数値計算アルゴリズムの紹介
SX-Aurora TSUBASA 高速化技法の基礎 【9月19日(火)】	性能測定や基礎的なチューニング手法の説明
ONION-object 入門 【9月20日(水)】	ONION-object（Cloudian製のHyperStoreで提供されるオブジェクトストレージ）の説明と利用方法
コンテナ入門 【9月22日(金)】	Singularityを利用したコンテナジョブの投入方法とカスタマイズ方法
並列プログラミング入門(OpenMP/MPI) 【9月25日(月)】	OpenMP、MPI、ベクトルプロセッサで利用できる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

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！