

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

2018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 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:64 VE:10
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 16.519PFLOPS

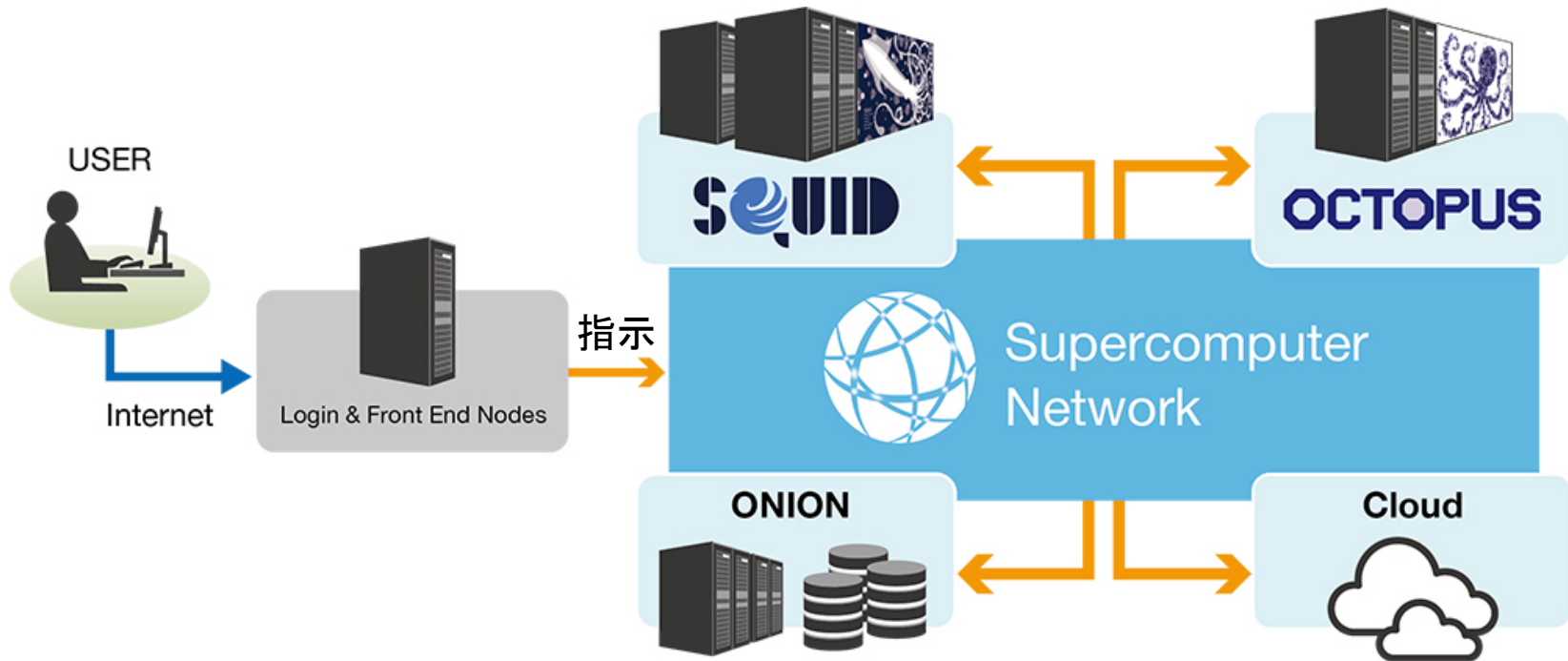
フロントエンドサーバ

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

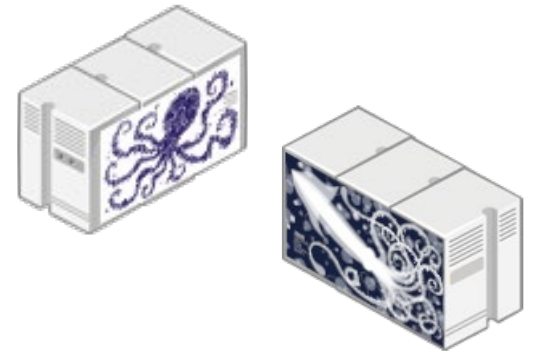


フロントエンド端末



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
SQUID : 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へのログインについて

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	Google Play Store
iOS	Microsoft Authenticator	Apple App Store
Windows	WinAuth	Github
macOS	Step Two	Apple App Store





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



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

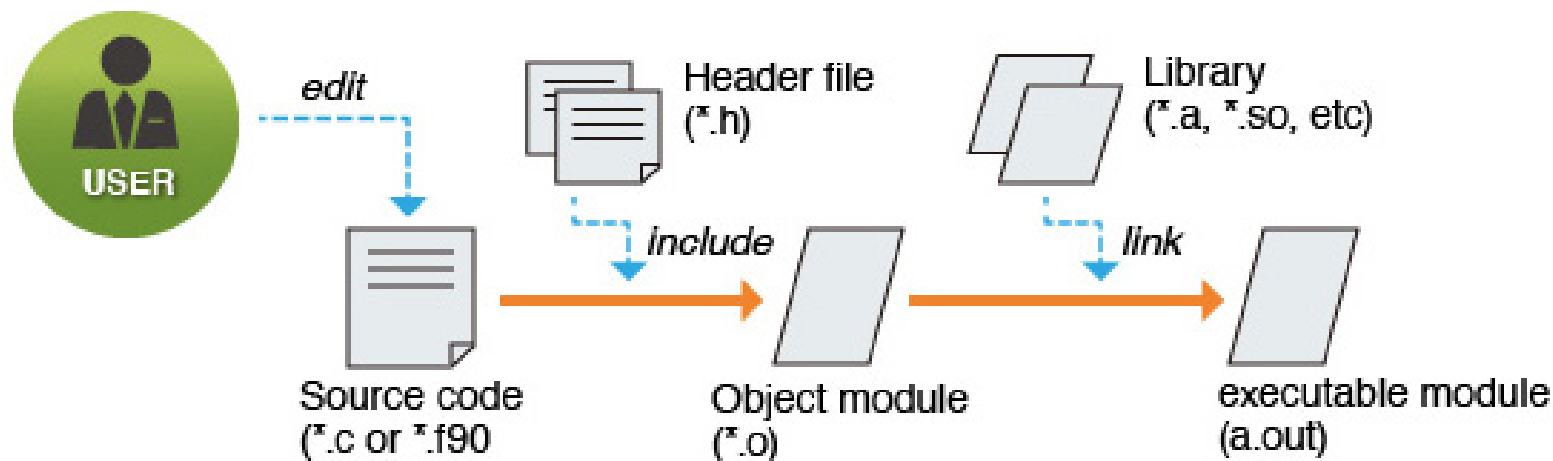
当センターの計算機で使用可能な主な言語

Fortran言語、C言語、C++言語

「プログラムの書き方」については
特に説明しません

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

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/2021	BaseGPU/2021	BaseVEC/2021

コマンド例: 汎用CPUノードを利用する場合

```
$ module load BaseCPU/2021
```

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

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

SQUIDでa.outという
プログラムを実行したい

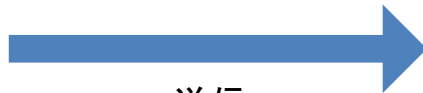
ジョブスクリプトを作成



ユーザー

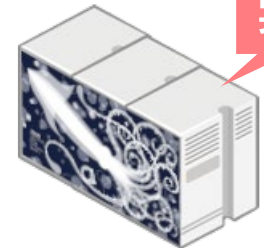


送信



ジョブ
スケジューラ

計算機が空き次第
実行指示



実行

ジョブの投入が
終われば
ログアウトしてもよい



終了次第、実行結果がファイル出力される

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/2021
```

```
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

NQSII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
NQSII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 GPUノード	120時間	3,072Core (24Core×128ノード)	24,320GB (190GB×128ノード)	128ノード
OCTPHI	Xeon Phiノード	120時間	2,048Core (64Core×32ノード)	6,080GB (190GB×32ノード)	32ノード
OCTMEM	大容量主記憶 搭載ノード	120時間	256Core (128Core×2ノード)	12TB (6TB×2ノード)	2ノード
DBG	汎用CPUノード GPUノード	10分	24Core (24Core×1ノード)	190GB (190GB×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
NQSII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	利用可能 メモリ	同時利用 可能ノード数	備考
SQUID	24時間	38,912Core (76Core×512ノード)	124TB (248GB×512ノード)	512ノード	
SQUID-H	24時間	38,912Core (76Core×512ノード)	124TB (248GB×512ノード)	512ノード	高優先度
SQUID-S	24時間	38Core (76Core×0.5ノード)	124GB (248GB×0.5ノード)	1ノード	ノード共有
DBG	10分	152Core (76Core×2ノード)	496GB (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NQSII 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/2021
```

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

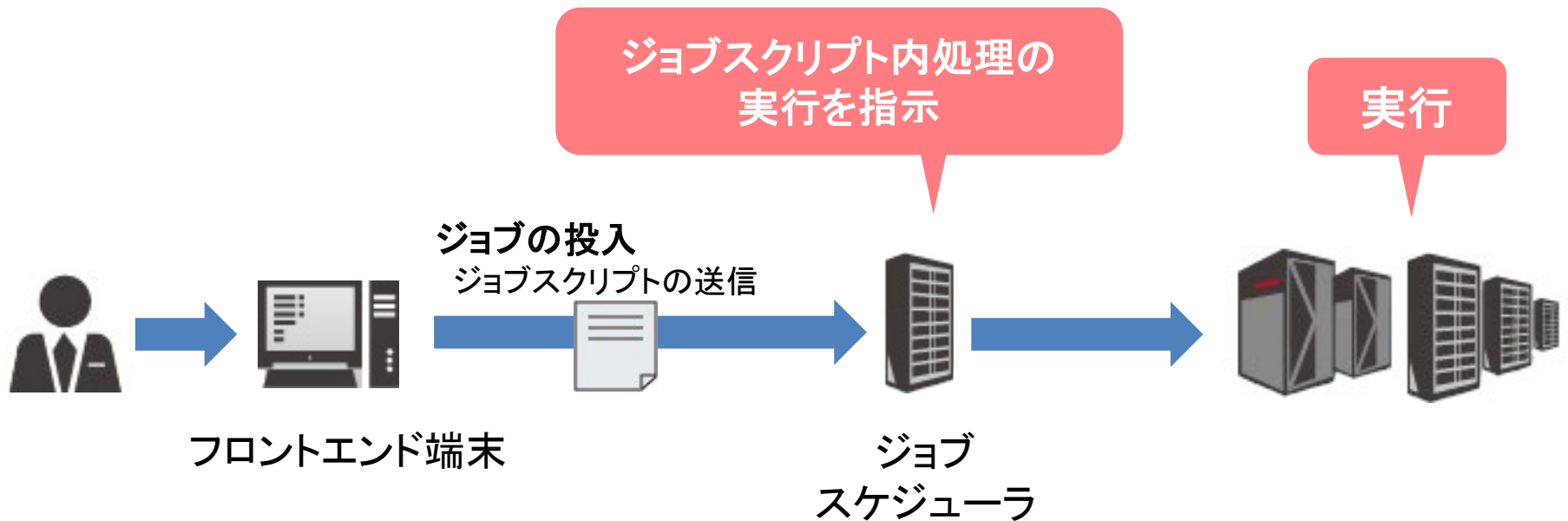
I. システムのご紹介

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

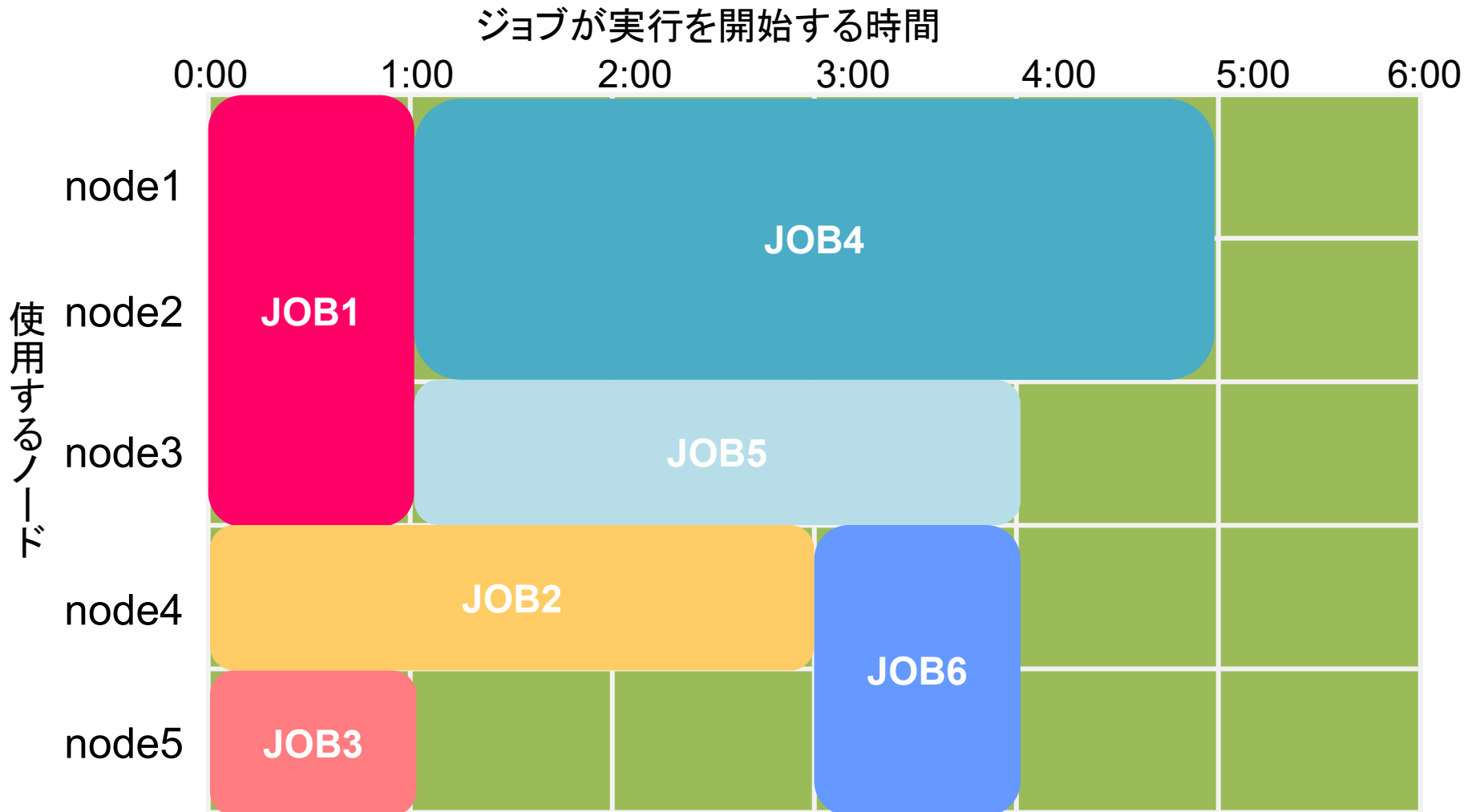
特徴

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当てる

スケジューラのイメージ



「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

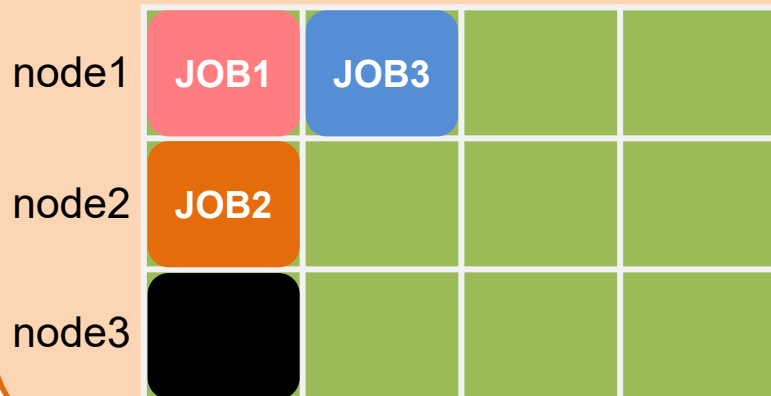
コマンド

\$ **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

ジョブの予約状況の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/

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

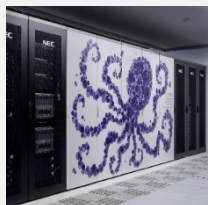
民間企業

利用負担金が必要になります

まずは無料利用をお試ください

3カ月間 以下の資源をご提供

無料



OCTOPUS

26 OCTOPUSポイント
+ ディスク 3TB

500 ノード時間

汎用
CPU
ノード

622 ノード時間

Xeon
Phi
ノード

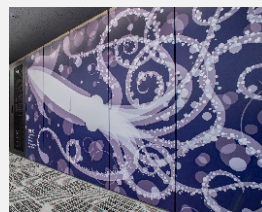
119 ノード時間

GPU
ノード

70 ノード時間

大容量
主記憶
搭載
ノード

※季節係数が1の場合



SQUID

75 SQUIDポイント
+ ディスク 5TB

81 ノード時間

GPU
ノード

500 ノード時間

汎用
CPU
ノード

132ノード時間

ベクトル
ノード

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/

までお気軽にご連絡ください！

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・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

「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)

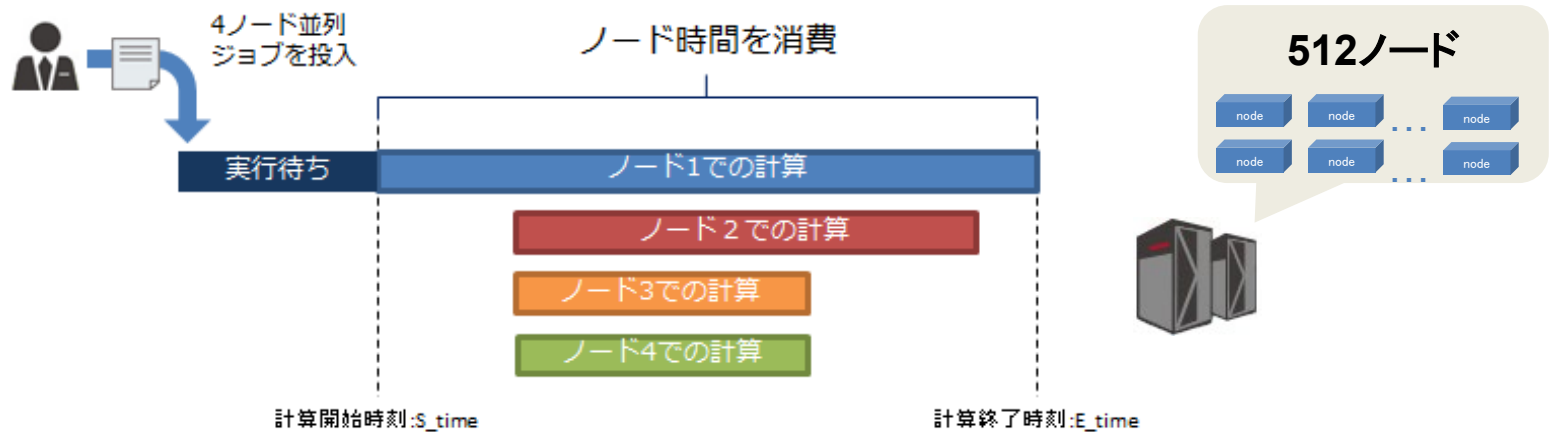
1ノードで3時間の計算	→	3ノード時間消費
30ノードで5時間の計算	→	150ノード時間消費
100ノードで1時間の計算	→	100ノード時間消費
1ノードで100時間の計算	→	100ノード時間消費

「ノード時間」の注意点



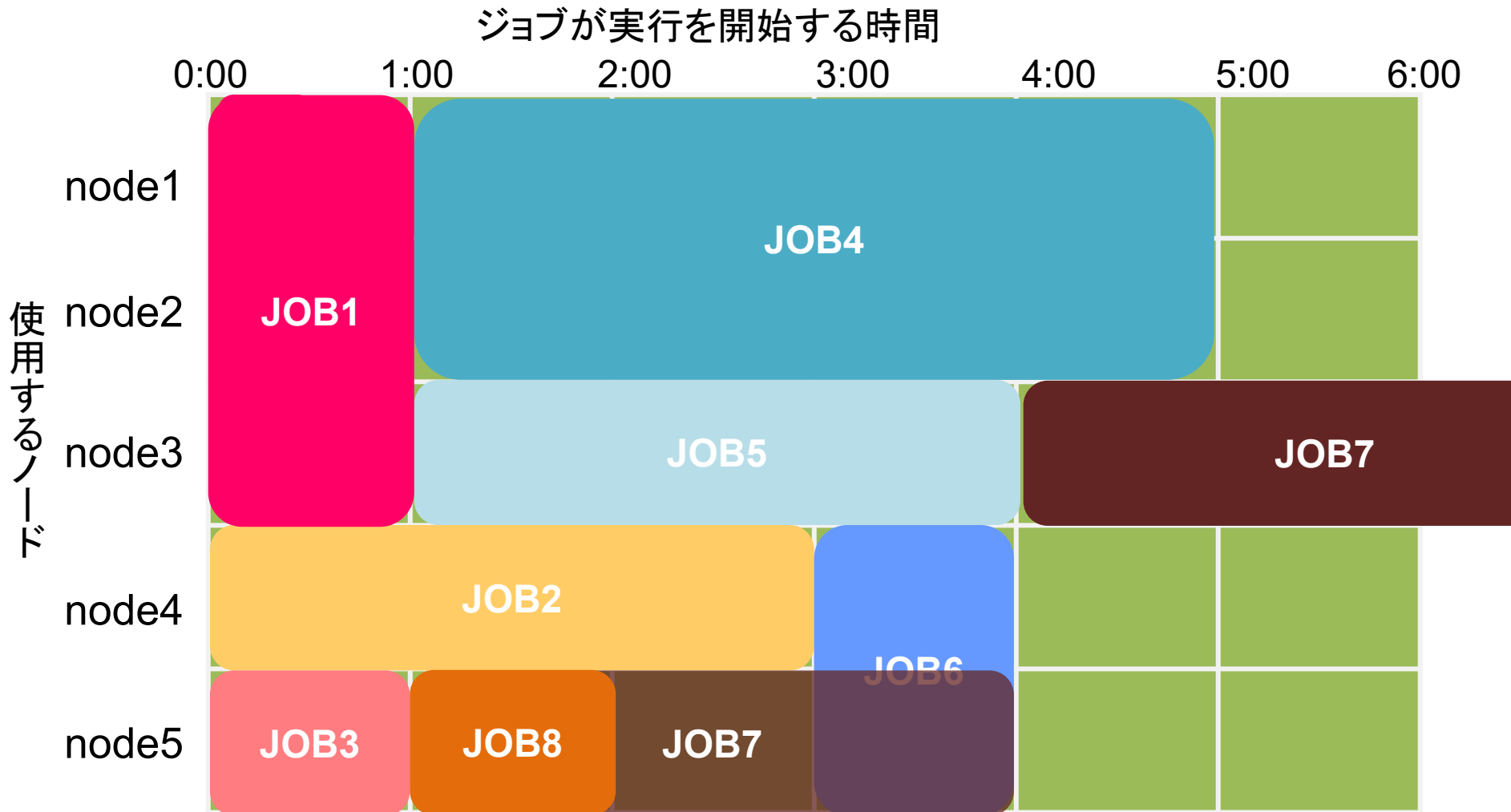
ノード内で使用するコアを限定しても、ノード時間は変わりません

「ノード時間」の注意点



ノード時間は4ノード × (計算終了時間 - 計算開始時間)です

スケジューラのイメージ



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

※OCTOPUSは今年度の受付を終了しました

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>

利用方法の解説

本項では共用計算機初心者を対象に
大規模計算機システムの利用方法を解説します

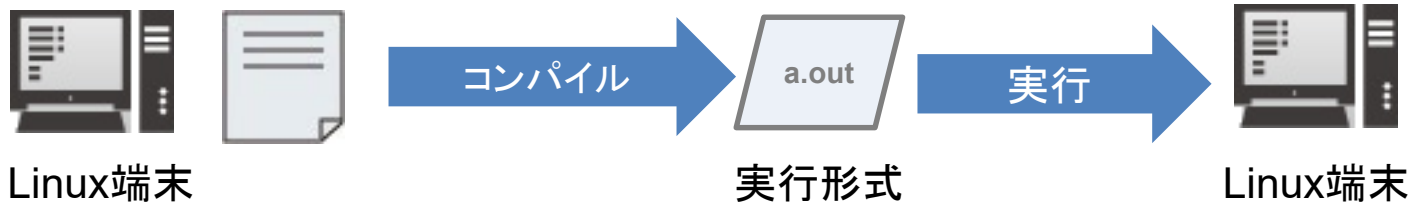
途中、スパコンを利用したデモを行います

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ください

コンパイルの種類

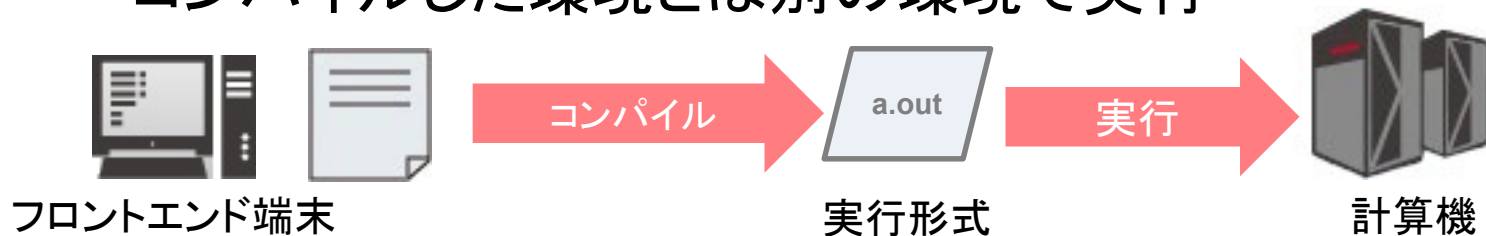
セルフコンパイル

コンパイルした環境と同じ環境で実行



クロスコンパイル

コンパイルした環境とは別の環境で実行



当センターでは「クロスコンパイル」を使用

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 sample.nqs

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 sample.nqs.oXXXXXX

\$ cat sample.nqs.eXXXXXX

※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

デモ2(ジョブスクリプト)

1. 演習用スクリプトを取得してください

```
$ cp /sc/cmc/apl/kousyu/nyumon/sample.nqs ~/
```

2. sample.nqsを元にSX-ACE用のジョブスクリプトを作成してください

```
$ emacs sample.nqs -nw
```

(参考) emacsの操作方法: 保存 ctrl-x → ctrl-s

終了 ctrl-x → ctrl-c

ジョブクラスはDBGを使用してください

デモ1(コンパイル)

1. 演習用プログラムの取得(コピー)

```
$ cp /sc/cmc/apl/kousyu/nyumon/sample.f ~/
```

2. sample.f をSX用にコンパイル

```
$ sxf90 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

より高度な利用に向けて

本日以降に実施予定の講習会

講習会名	日時	場所
SX-ACE 高速化技法の基礎	9月16日(水) 13:30 - 16:30	オンライン配信
並列コンピュータ高速化技法の基礎 (OCTOPUS向け)	9月17日(木) 13:30 - 15:30	オンライン配信
並列プログラミング入門(MPI)	9月24日(木) 13:30 - 17:00	オンライン配信
OpenMP入門(仮)	9月30日(水) 13:30 - 14:30	オンライン配信