第138回お試しアカウント付き講習会
「スーパーコンピューター超入門」
2020年9月18日

講義編

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

2020/9/23 v2.0（修正版）

本講習会の内容

1. 注意事項
2. 本講習会での質問の仕方、ZoomおよびSlackの使い方
3. スパコンの特徴、スパコンでできること。
4. スパコンの使い方（座学）
   - SSH、Linux、CLI、モジュール、ジョブスケジューラ

5. 実習
   1. スパコンへのログイン
   2. プログラムのコンパイル、実行
   3. 並列プログラムのコンパイル、実行
   4. 結果のファイル転送

6. スパコンの利用申請

講習会の注意事項

- 1か月利用可能なアカウントが配布されます。
  - 利用規定に記載の用途以外に使用しないでください。
    - ✓ 研究、教育、社会貢献のために使用
  - 1か月経過後(10/19)にはアカウントが削除されるため、ファイルも削除されます。
  - アカウントの継続利用のためには、講習会を最後まで受講いただく必要があります。
- 企業の方がスパコンの利用申請をする場合、センターが開催するいずれかの講習会への参加が要件となります。
  - 講習会の最後までの受講が申し込みの要件となります。
- 本講習会に関する質問は
  tut138@cc.u-tokyo.ac.jp
  に送付ください。センターのスパコン相談窓口等には送付しないようご注意ください。

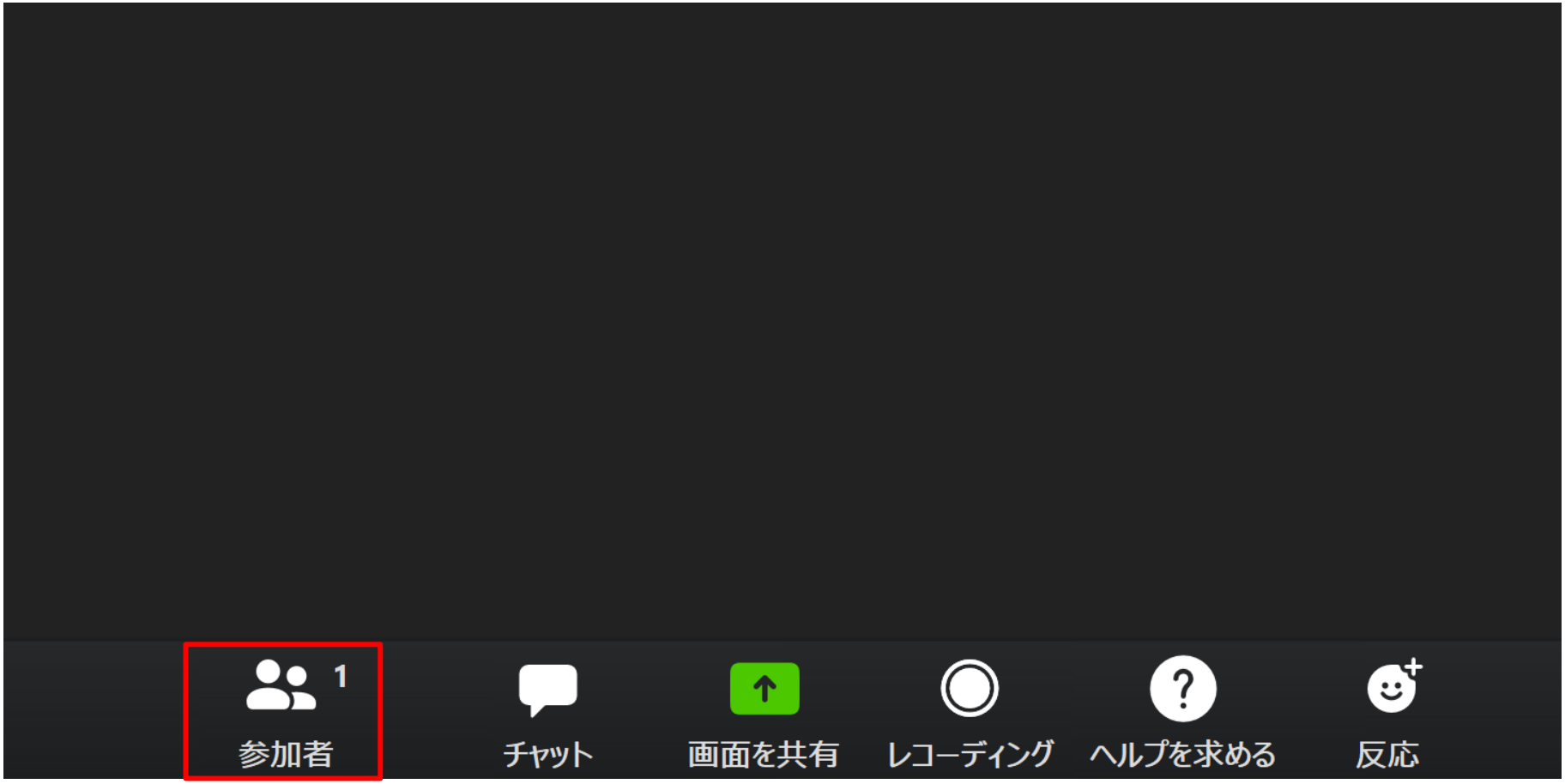
本講習会の内容

1. 注意事項
2. 本講習会での質問の仕方、ZoomおよびSlackの使い方
3. スパコンの特徴、スパコンでできること。
4. スパコンの使い方（座学）
   - SSH、Linux、CLI、モジュール、ジョブスケジューラ
5. 実習
   1. スパコンへのログイン
   2. プログラムのコンパイル、実行
   3. 並列プログラムのコンパイル、実行
   4. 結果のファイル転送
6. スパコンの利用申請

質問の方法

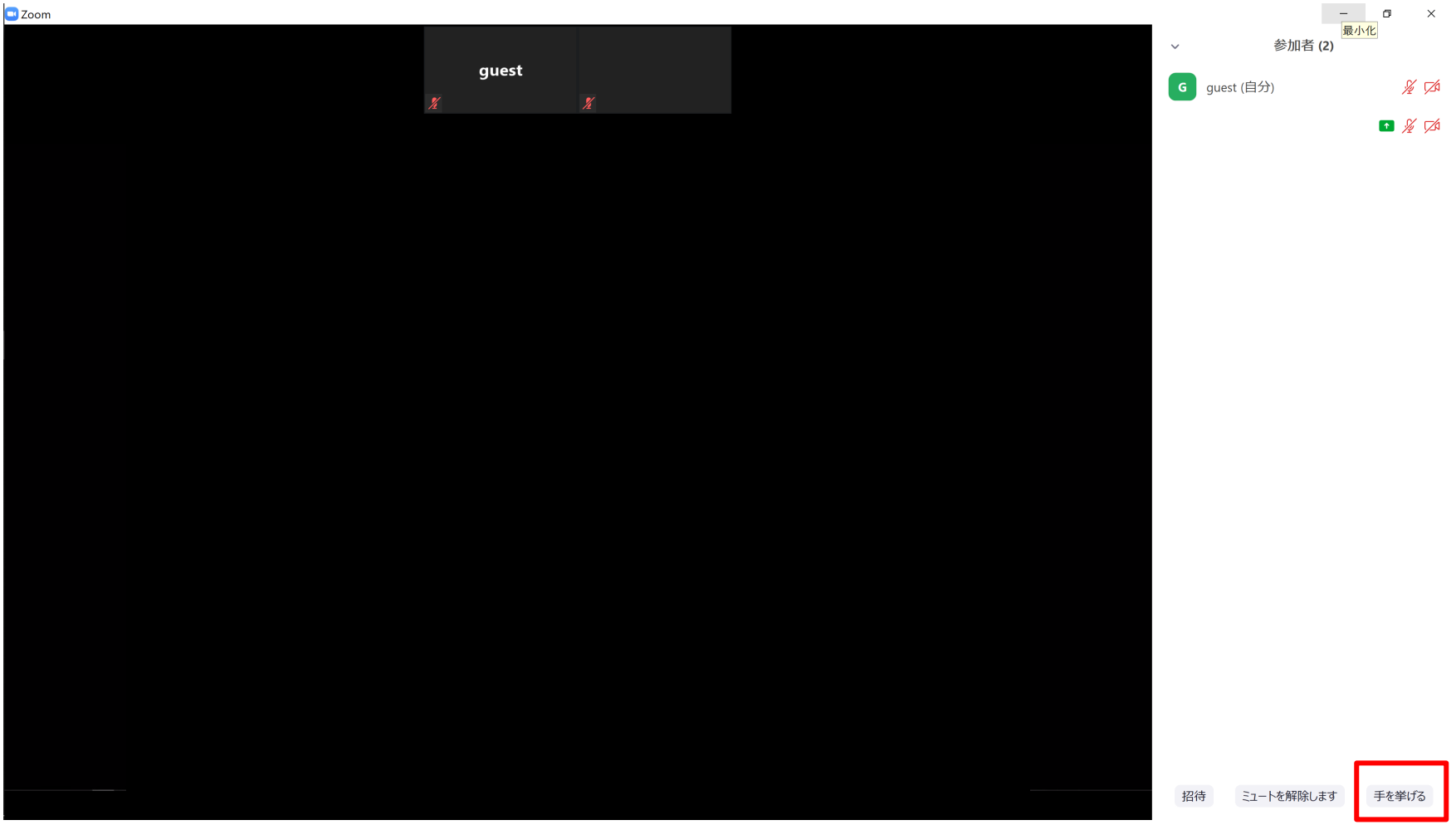
- 講義中の質問方法
  - Zoomの挙手後の発言
  - Slackへの書き込み
- 実習中の質問方法
  - Zoomの挙手後の発言
  - Slackへの書き込み
  - Zoomの“ヘルプを求める”機能の利用
    - ✓ 画面を共有してのヘルプ

Zoomでの挙手 1/3



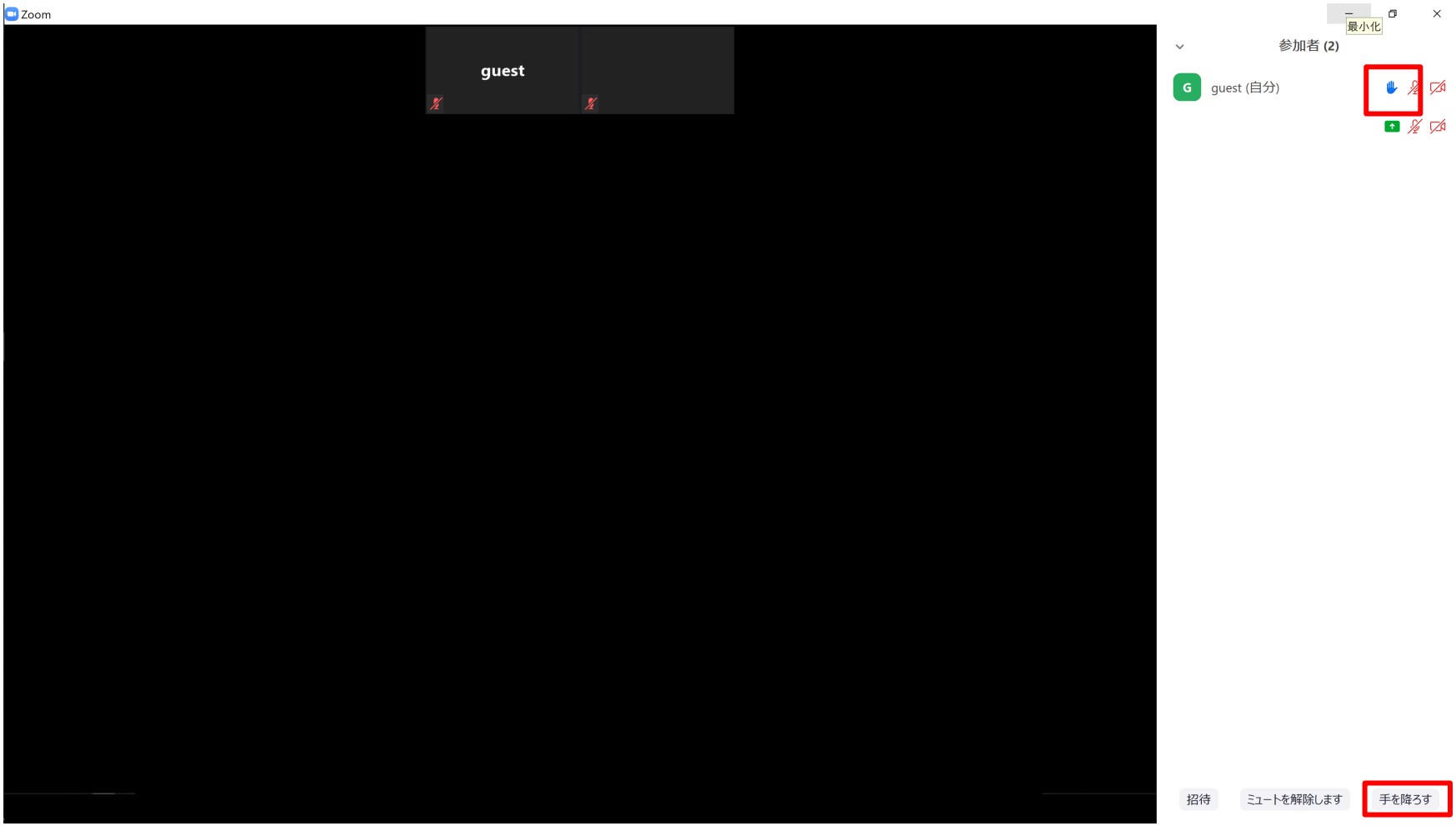
- “参加者”をクリック
- 画面の右側に参加者リストが出現

Zoomでの挙手 2/3



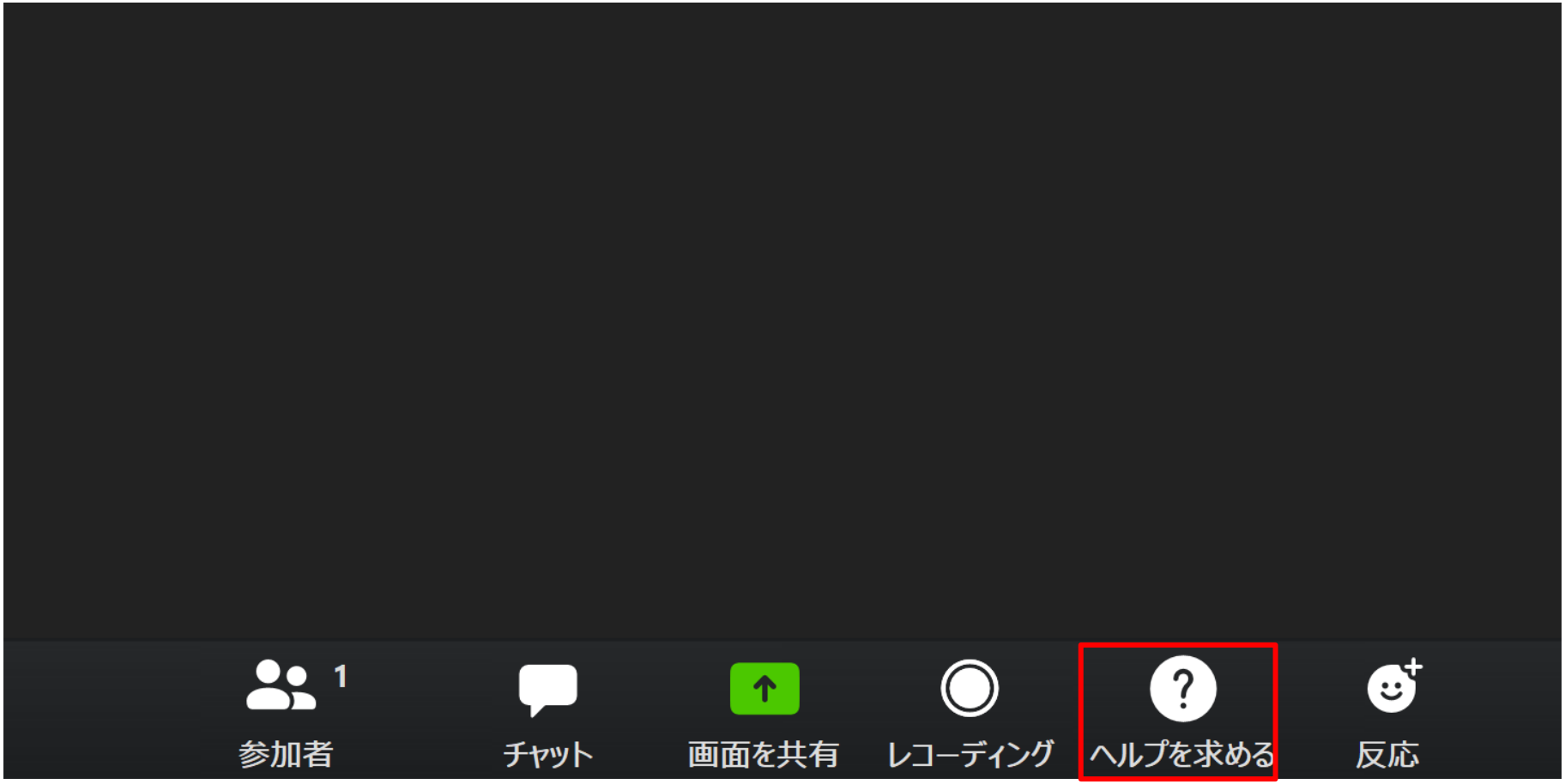
■ 右下の“手を挙げる”をクリック

Zoomでの挙手 3/3



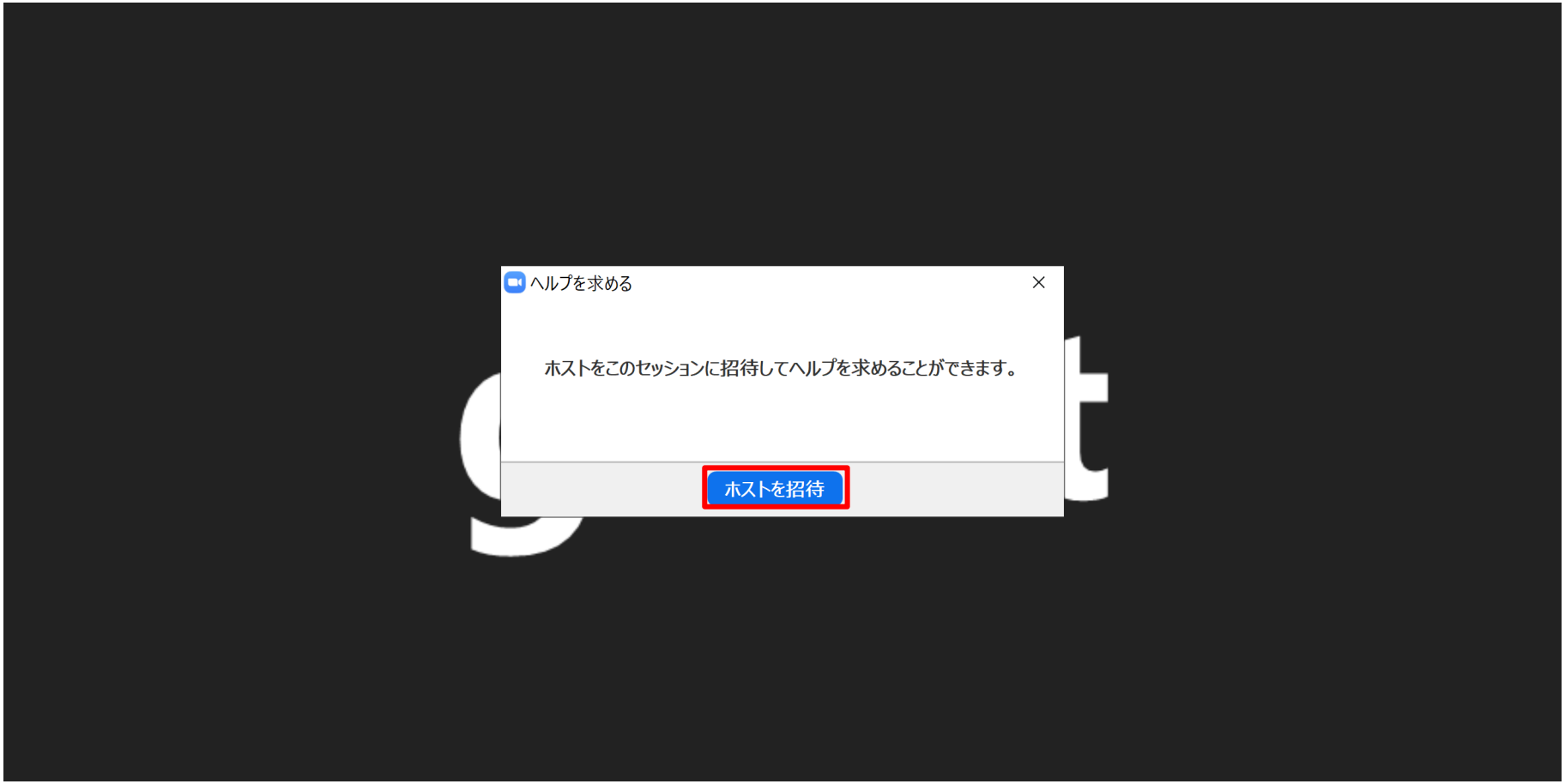
- ユーザー名の横に手のアイコンが出現
- “手を降ろす”をクリックでキャンセル

Zoomでヘルプを求める



- 実習時、講師のヘルプを求めるために使用（受講者の画面を共有します。）
- Zoom画面の下の一覧の中の“ヘルプを求める”をクリック

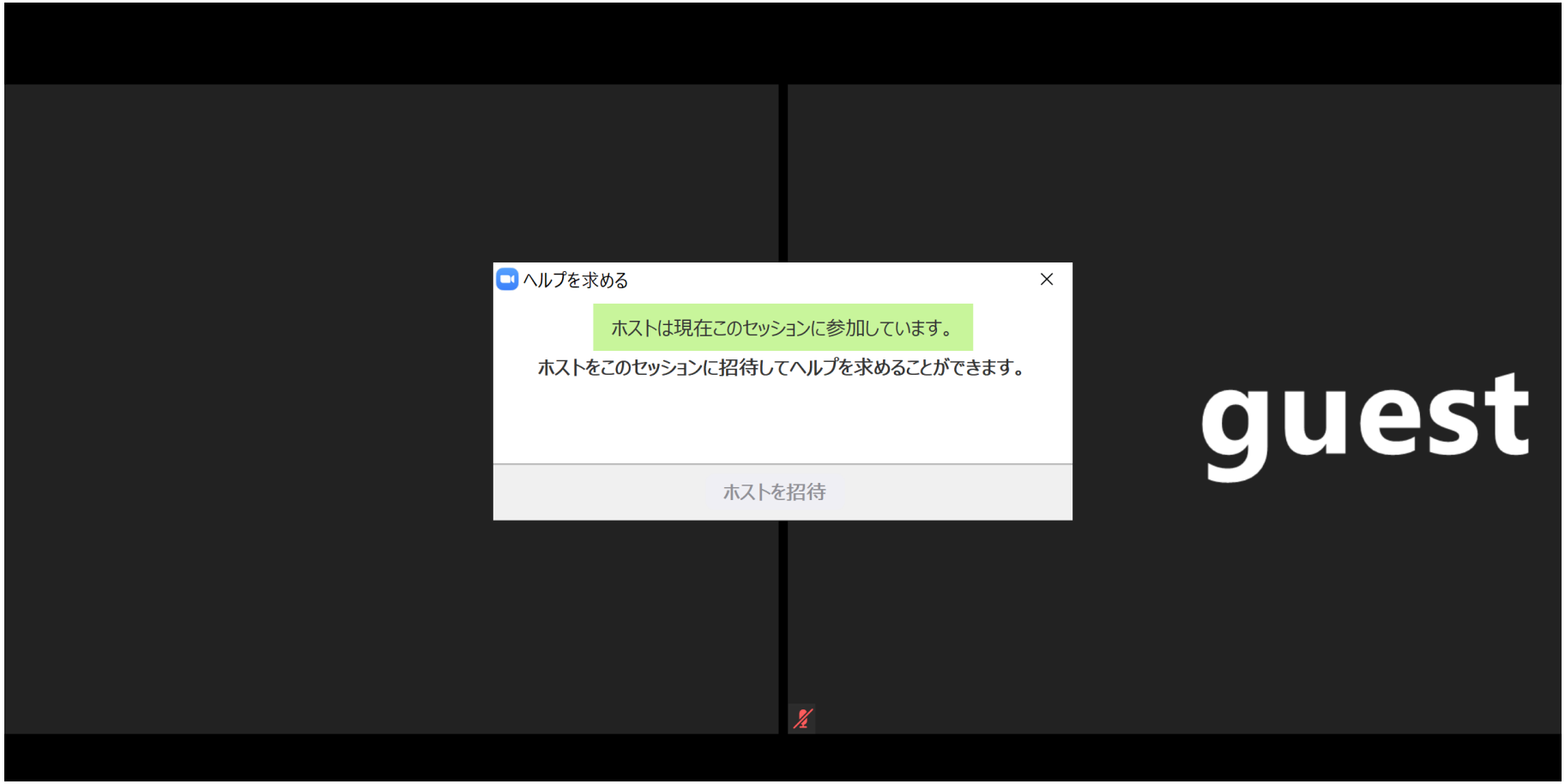
ブレークアウトルーム待機中



- ”ホストを招待“をクリック
- ホストの承認待ちに移行

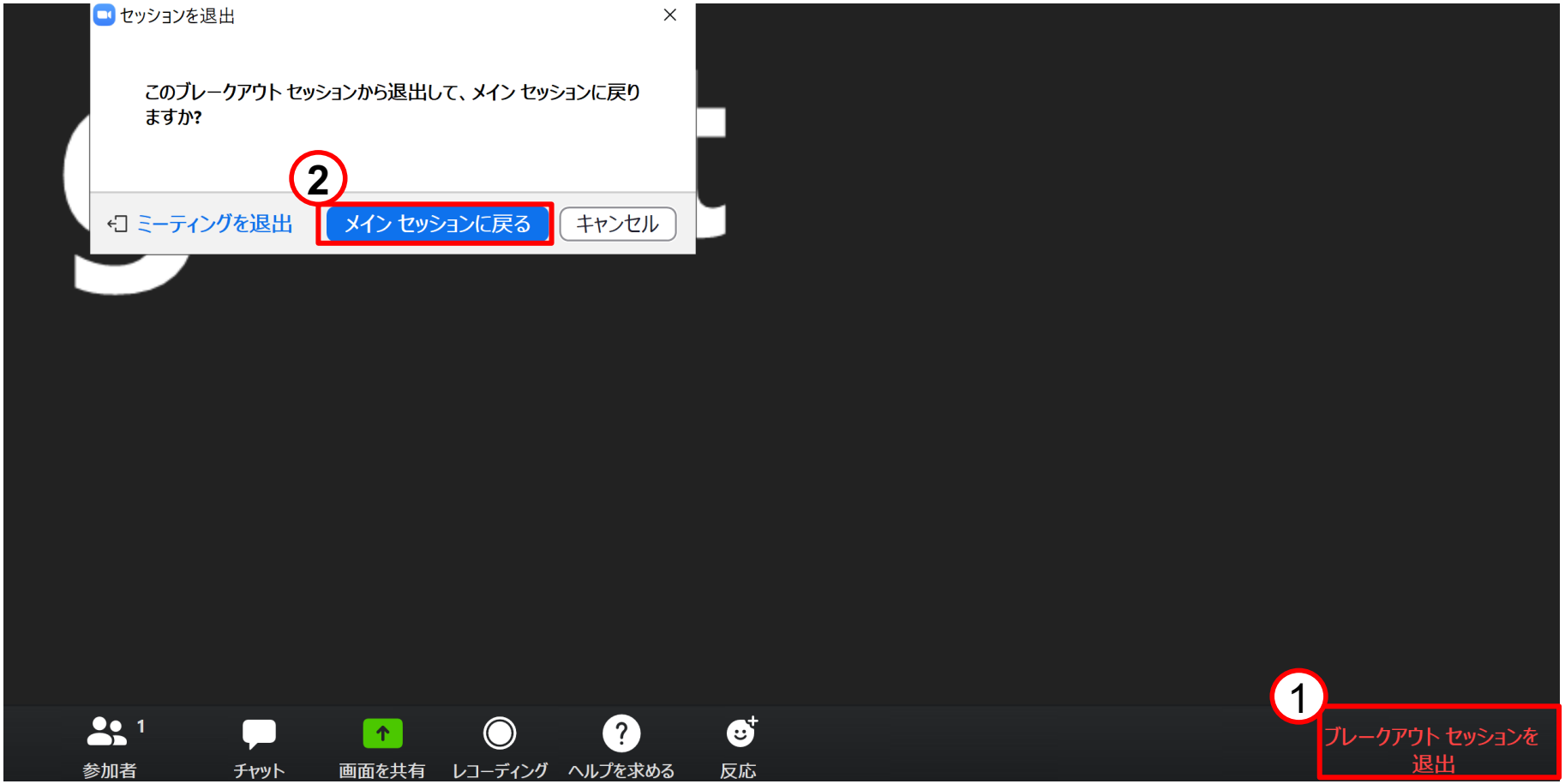
注：すぐにブレイクアウトルームには入りません。（他の受講者のヘルプ中）

ブレイクアウトルーム



■ この状態なら講師と会話可能

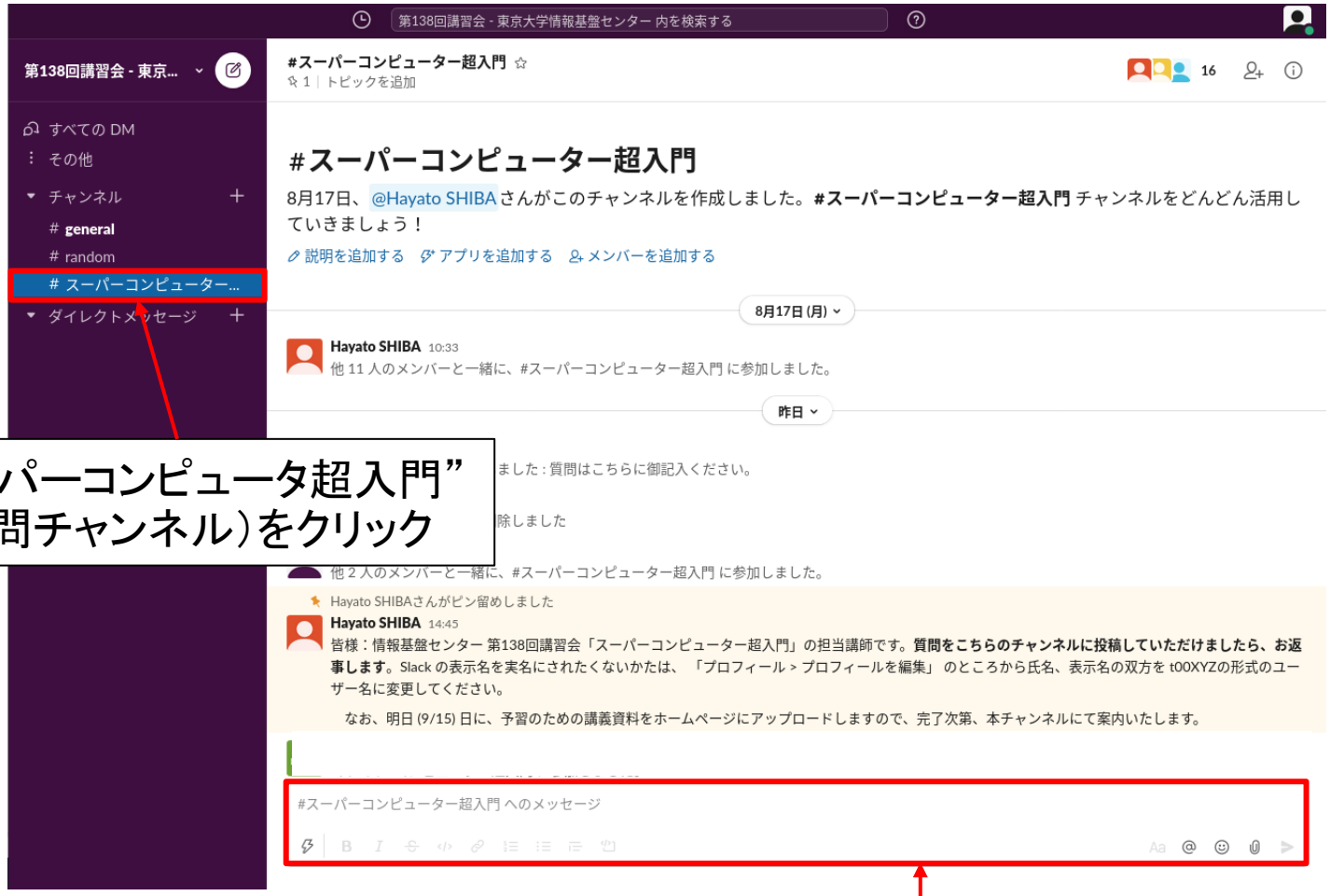
ブレークアウトルームからの退出



- 一定時間経過後に自動的に退出します。
- 任意に退出したい場合の手順になります。

Slackの使い方

質問にはSlackもご利用ください。

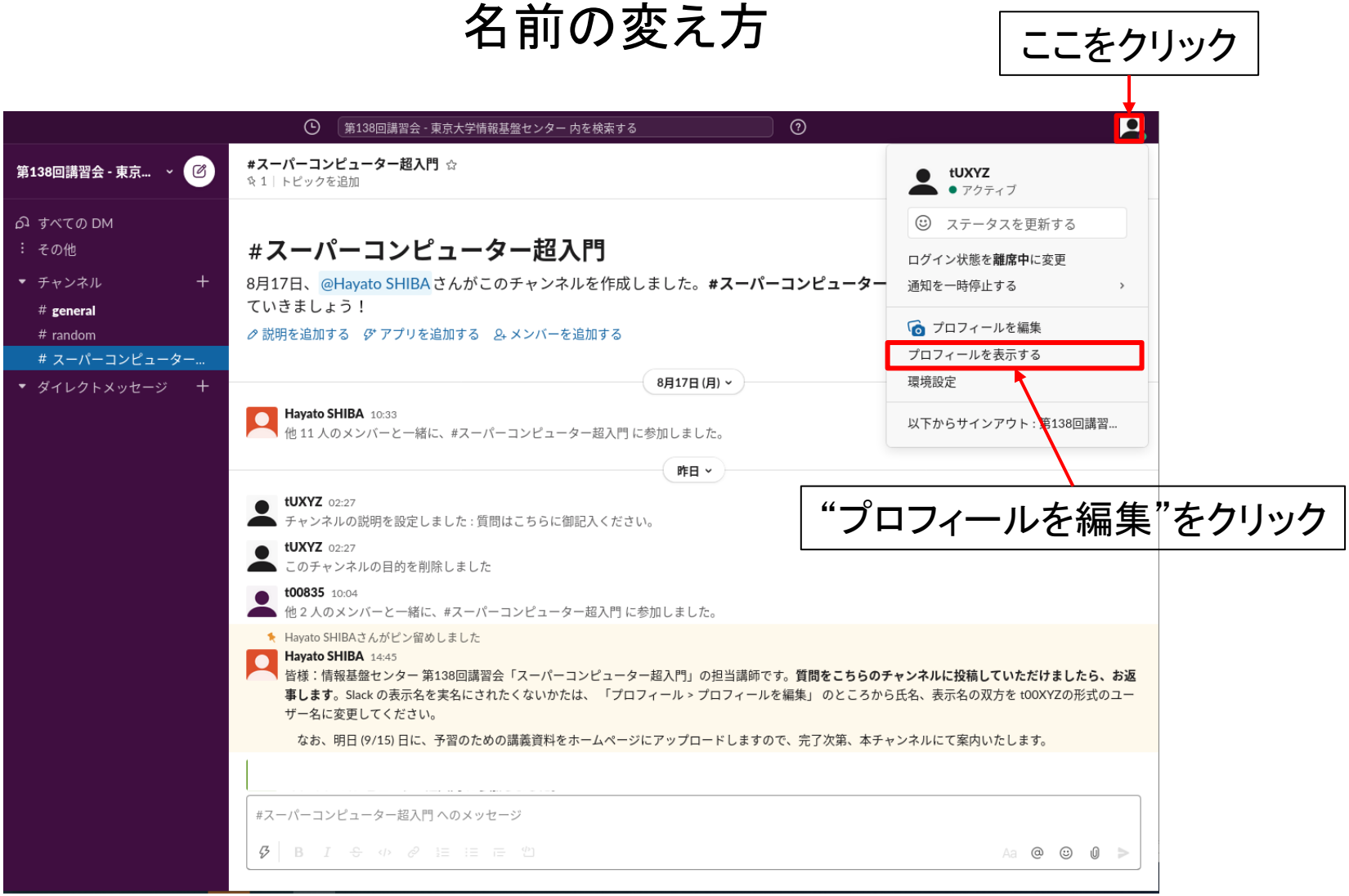


“スーパーコンピュータ超入門”（質問チャンネル）をクリック

ここに質問を書いて Ctrl+Enter

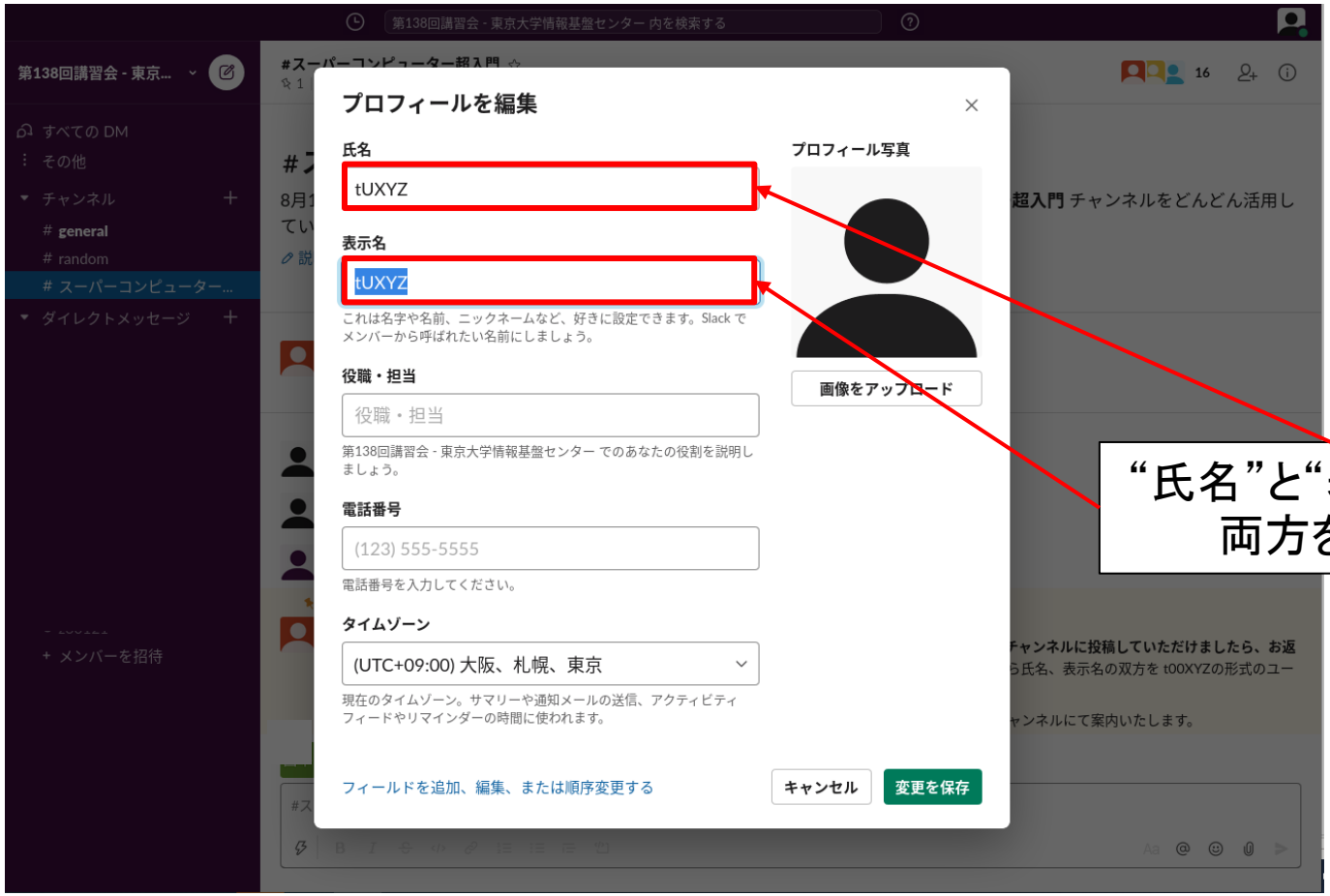
Slackでの名前の変え方 1/2

名前の変え方



Slackでの名前の変え方 2/2

名前の変え方



本講習会の内容

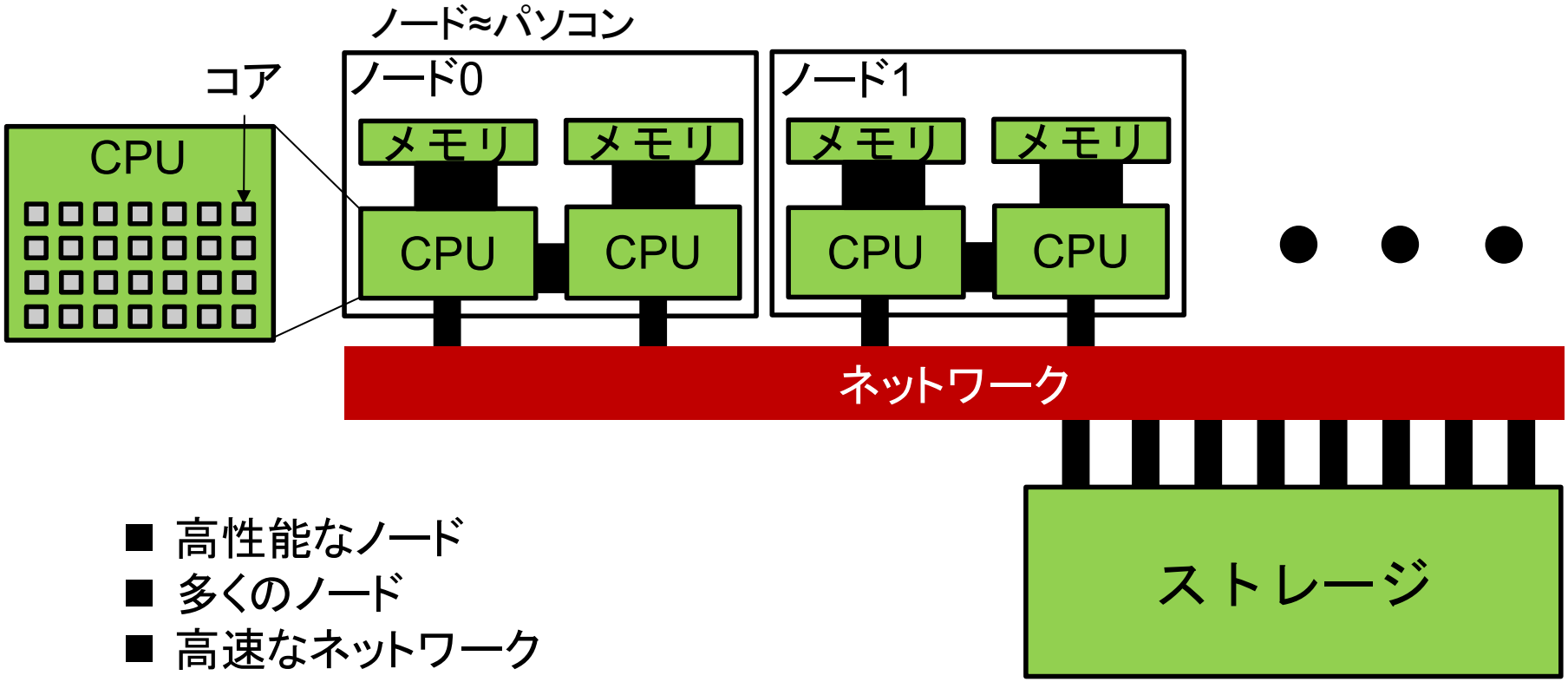
1. 注意事項
2. 本講習会での質問の仕方、ZoomおよびSlackの使い方
3. スパコンの特徴、スパコンでできること。
4. スパコンの使い方（座学）
   - SSH、Linux、CLI、モジュール、ジョブスケジューラ

5. 実習
   1. スパコンへのログイン
   2. プログラムのコンパイル、実行
   3. 並列プログラムのコンパイル、実行
   4. 結果のファイル転送

6. スパコンの利用申請

スパコンとパソコンの違い

高い性能をもつ大規模計算機



- 高性能なノード
- 多くのノード
- 高速なネットワーク
- 大容量かつ高速なストレージ

今回ご利用いただくOBCXを例に説明

ノードの違い

基本的に、コア数が多く、メモリの転送速度が速い

		Mac book	Mac Pro	OBCX
CPU	コア数	8	28	56 (2ソケット)
	周波数 [GHz]	2.3	2.5	2.7
	理論演算性能[TFlops]	0.59	2.24	4.84
Memory	容量 [GByte]	16	1500	192
	転送速度 [GByte/秒]	42.6	140.8	281.6

数字が大きいほど良い。

OBCXのノード

ものによってはGPU付き

数(ノード数)の違い

たくさんのパソコンを並列に使用する

スパコンが大規模といわれる所以

×4

×17

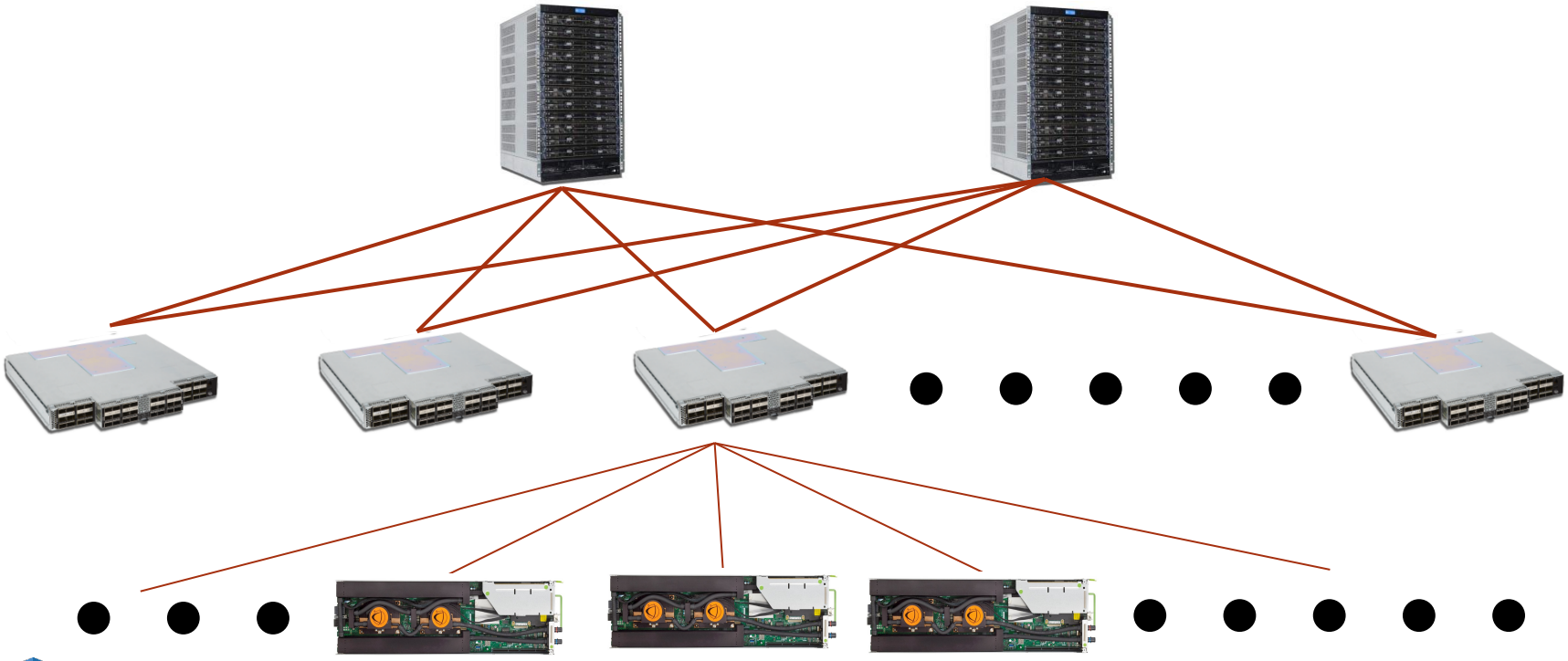
×21

全部で1,368台

ネットワークの違い

ノード数が多い(1,368)ため、ネットワークは非常に高速

	無線 (Wifi ac)	有線 (Gigabit Ether)	Omni-path (OBCX)
転送速度 [Gbit/秒]	6.9	1	100



ストレージの違い

全体からのアクセスに対応するため、大容量かつ高速

OBCXのストレージ(Luster)の仕様

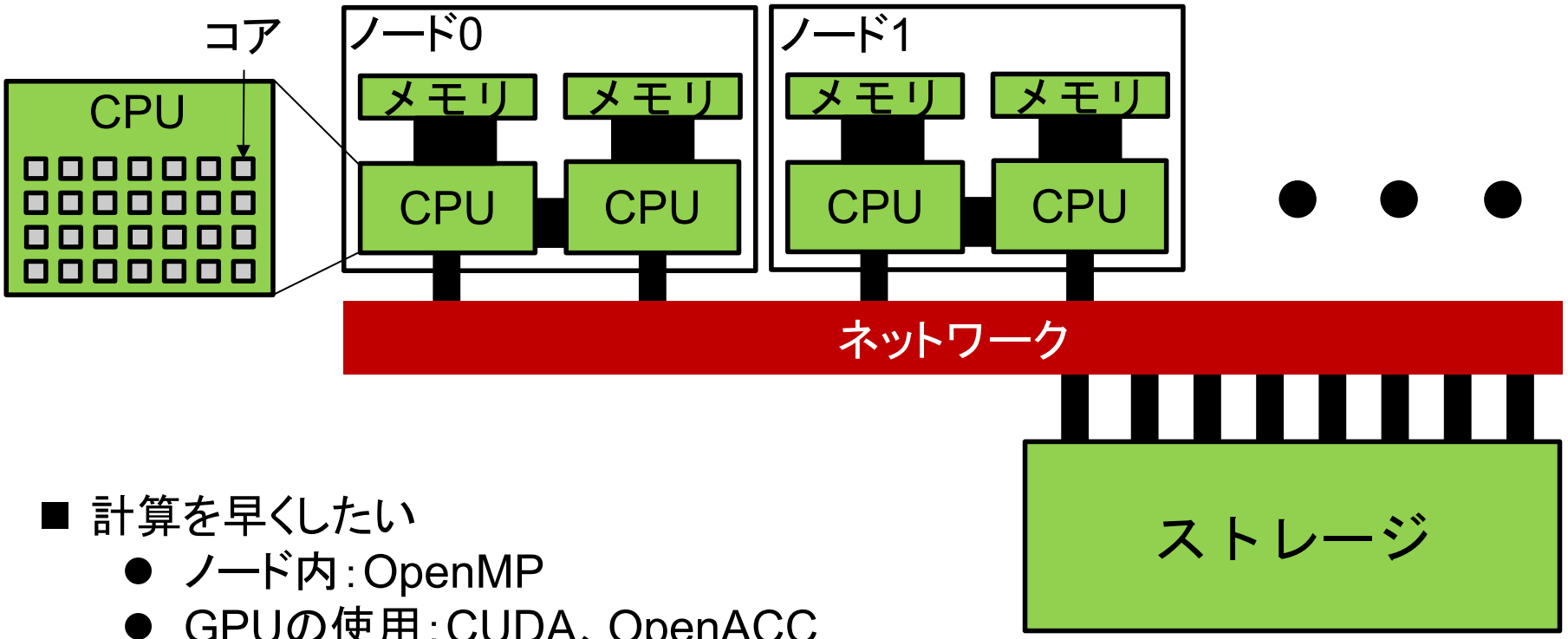
- 容量：12.4PByte (12400TByte)
- 構成：HDD 1,360本
- 転送速度：193.9GByte/秒

HDD,SSDの転送速度は
HDD ：0.1～0.2GByte/秒
SSD：4～5GByte/秒

*2020年9月現在

スパコンを効率的に使うためには

目的とスパコンの構成を把握し、適切な最適化を考える。



- 計算を早くしたい
  - ノード内：OpenMP
  - GPUの使用：CUDA、OpenACC
  - 複数ノードの使用：MPI
- I/Oを早くしたい：MPI (MPI_IO)
- 大規模な問題を解きたい：MPI

OBCX ハードウェア仕様

計算ノード

Chassis: PRIMERGY CX400 M4 x342 <4node / Chassis>
Node: PRIMERGY CX2550 M5 x1,240, CX2560 M5 x128

x1,368 node

全体性能

理論演算性能:	6.61PF
主記憶容量:	256.5TiB
メモリバンド幅:	385.1TB/s
ラック数:	21ラック
SSD搭載:	128ノード

ノード単体

理論演算性能: 4.8384 TF
手記憶容量: 192GiB
メモリバンド幅: 281.6GB/s

計算ノード間ネットワーク (Omni-Path Architecture)
通信性能 100Gbps

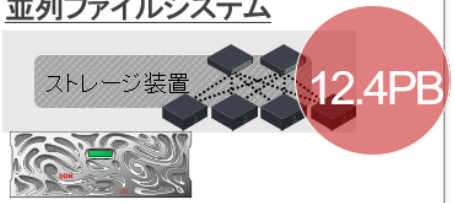
ログインノード

FUJITSU Server
PRIMERGY CX2560 M5 x 10

管理サーバ群

FUJITSU Server
PRIMERGY RX2530 M4 x 15
(ジョブ、運用、認証、Web、
セキュリティログ保存)

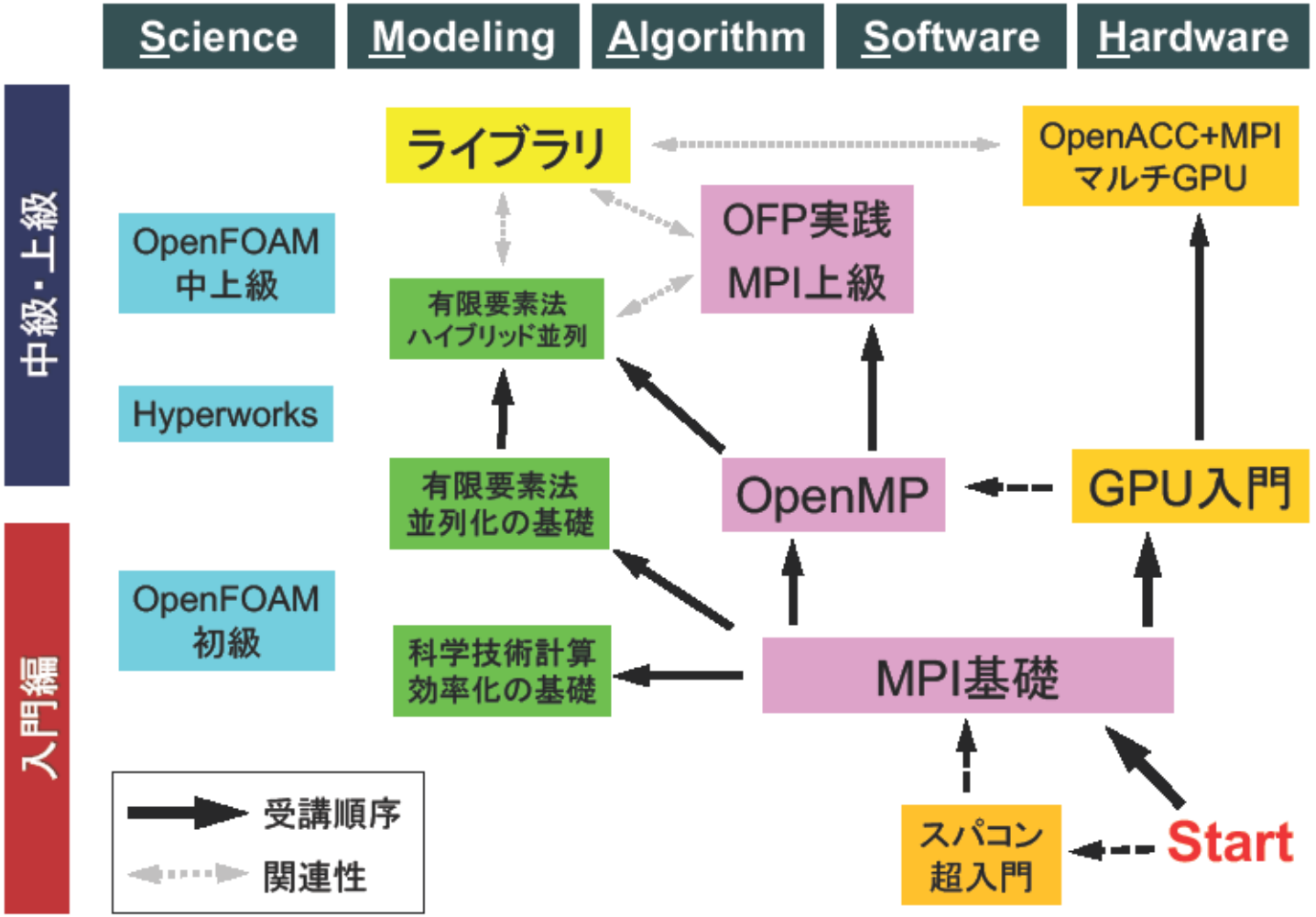
並列ファイルシステム



ストレージ装置：DDN ES18KE x2セット
ファイルシステム：DDN ExaScaler
(Lustreベースファイルシステム)

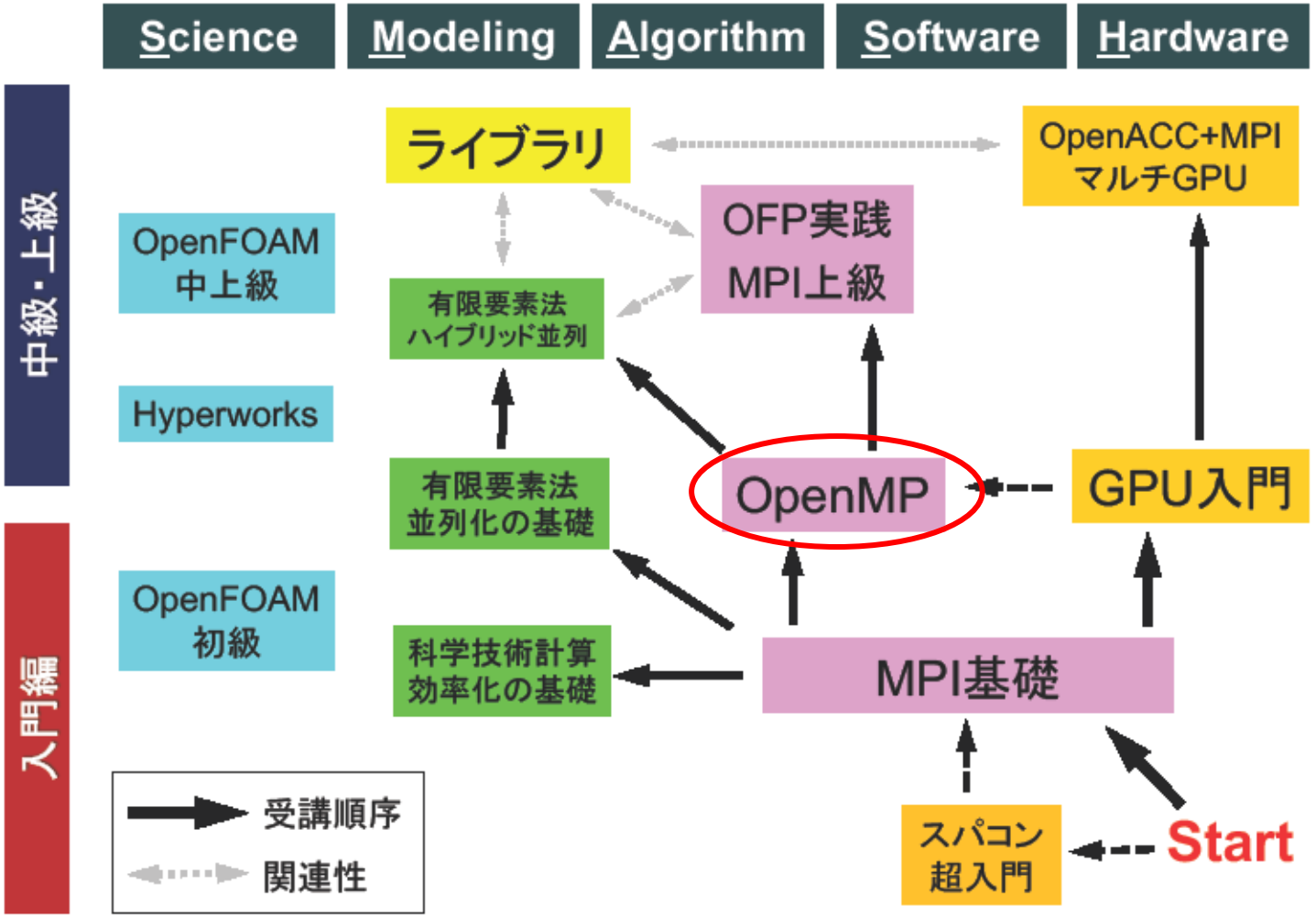
スパコン向け最適化に関する講習会

目的に応じて受講をご検討ください。



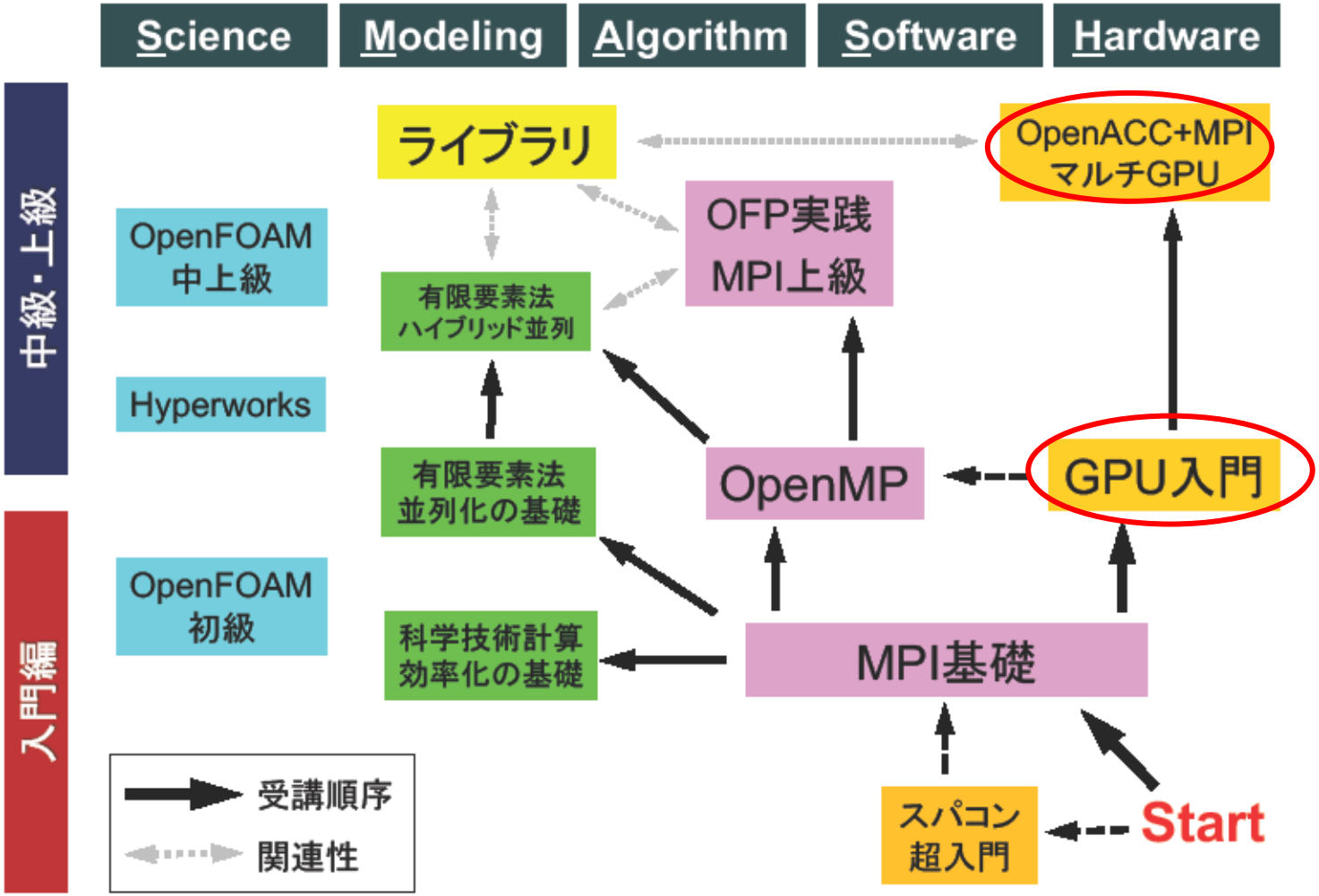
スパコン向け最適化に関する講習会

ノード内並列（OpenMP）に関する講習会



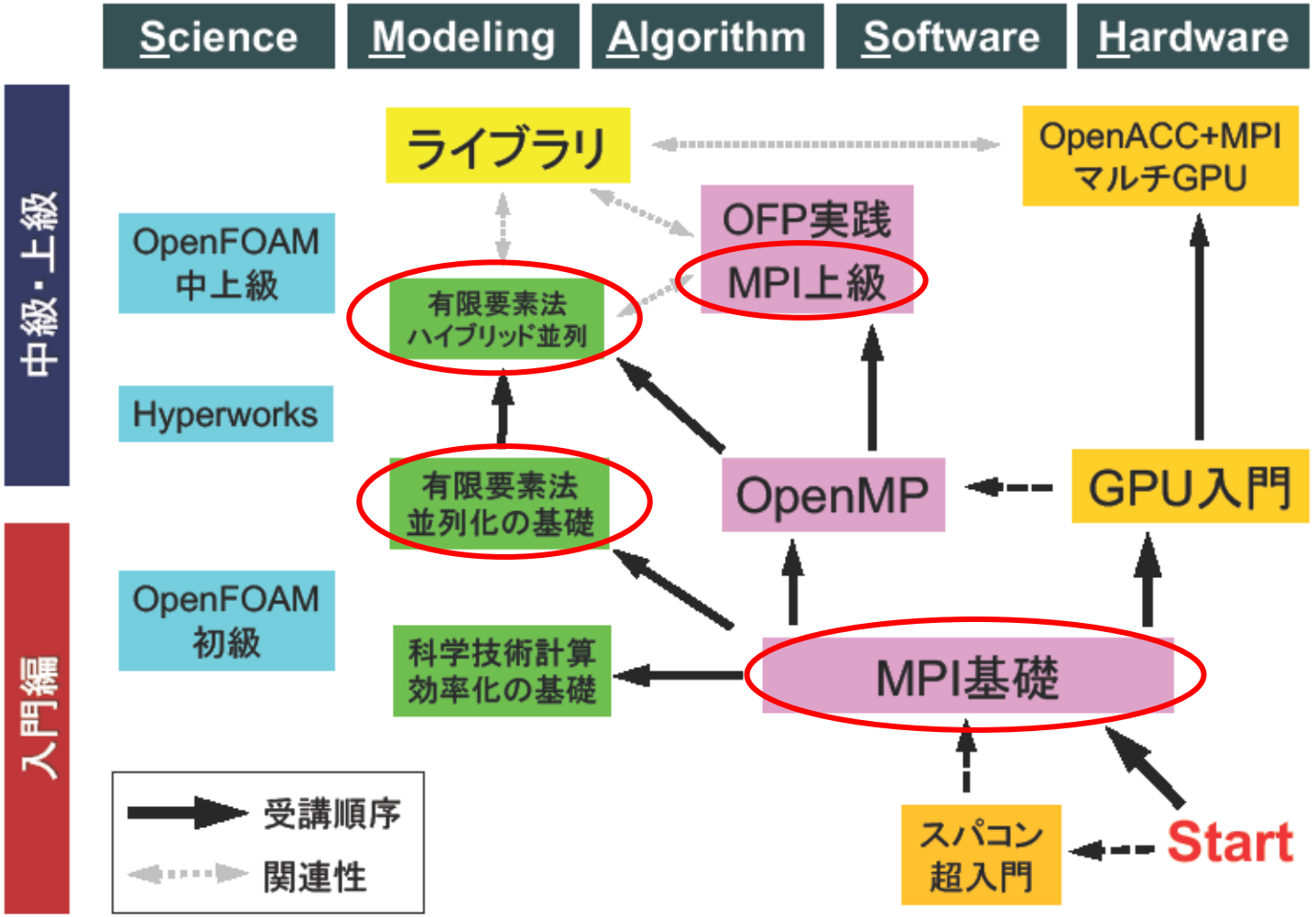
スパコン向け最適化に関する講習会

GPUを使用する講習会



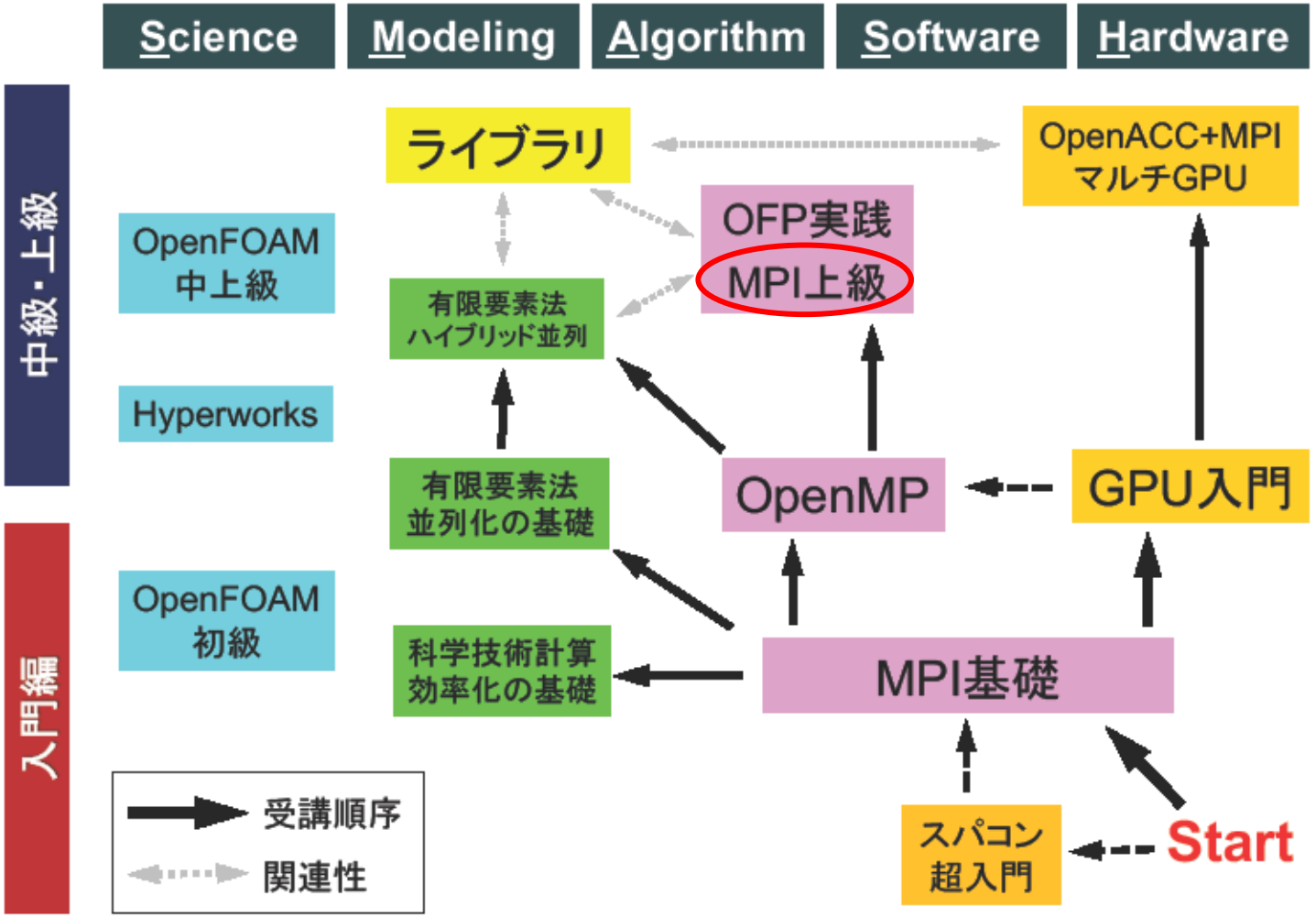
スパコン向け最適化に関する講習会

ノード間並列（MPI）に関する講習会



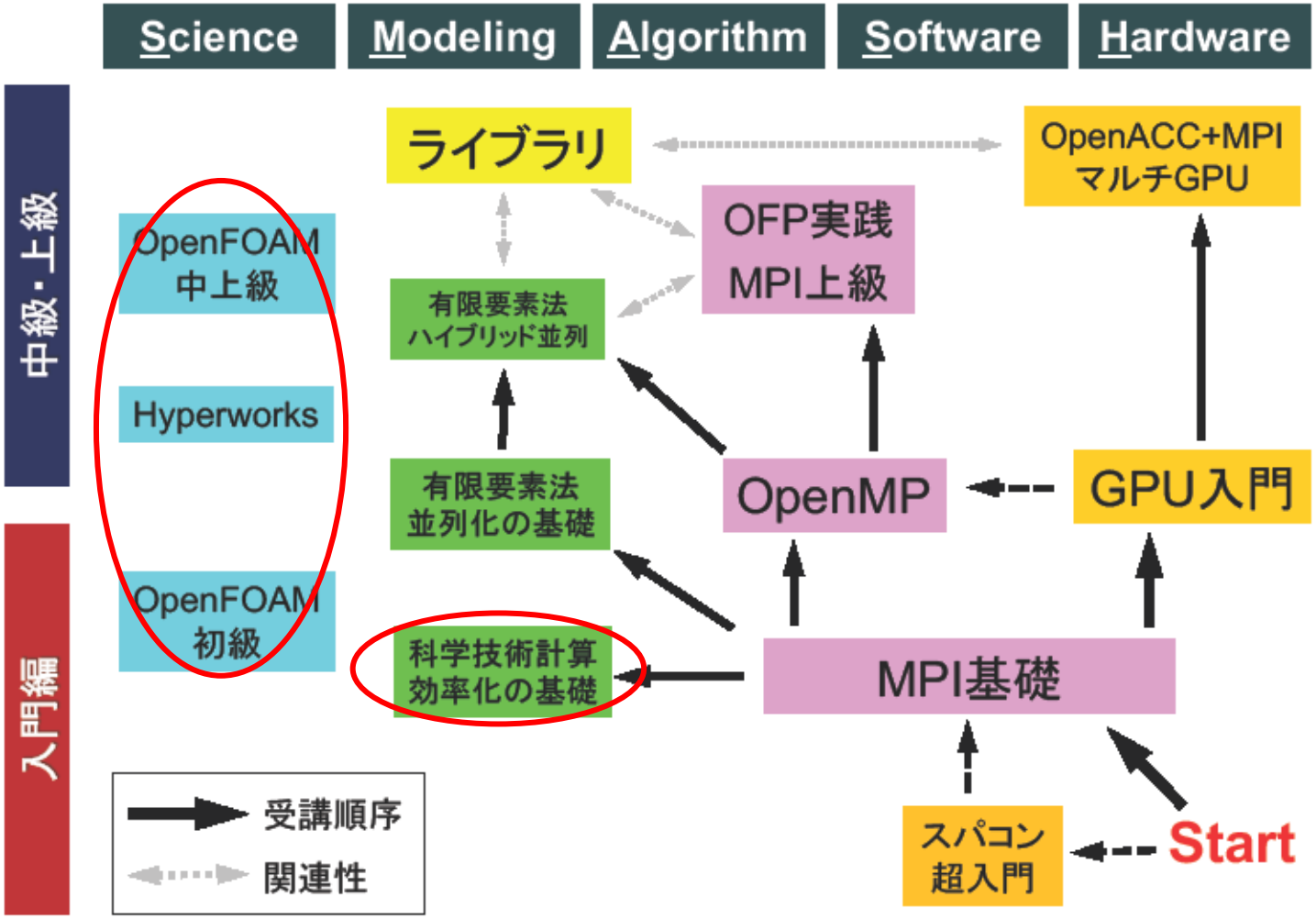
スパコン向け最適化に関する講習会

I/Oの高速化（MPI）に関する講習会



スパコン向け最適化に関する講習会

ライブラリやツール、ソフトウェアの使用に関する講習会

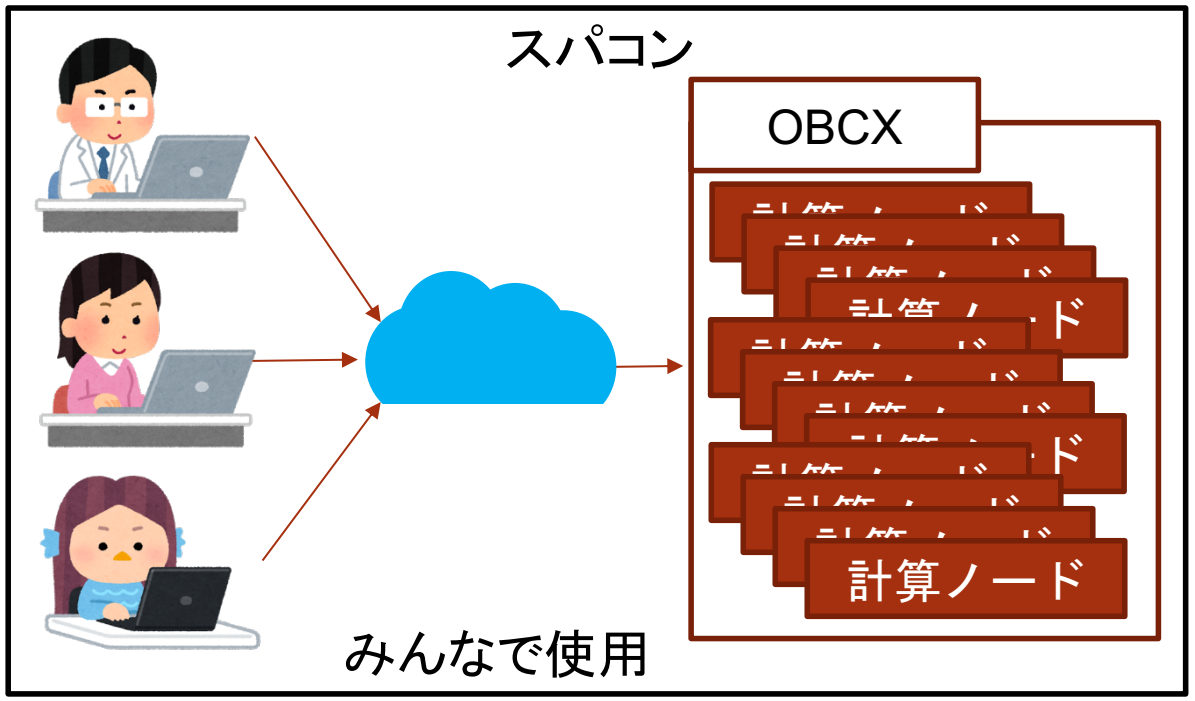


本講習会の内容

1. 注意事項
2. 本講習会での質問の仕方、ZoomおよびSlackの使い方
3. スパコンの特徴、スパコンでできること。
4. スパコンの使い方（座学）
   - SSH、Linux、CLI、モジュール、ジョブスケジューラ
5. 実習
   1. スパコンへのログイン
   2. プログラムのコンパイル、実行
   3. 並列プログラムのコンパイル、実行
   4. 結果のファイル転送
6. スパコンの利用申請

スパコンの使い方

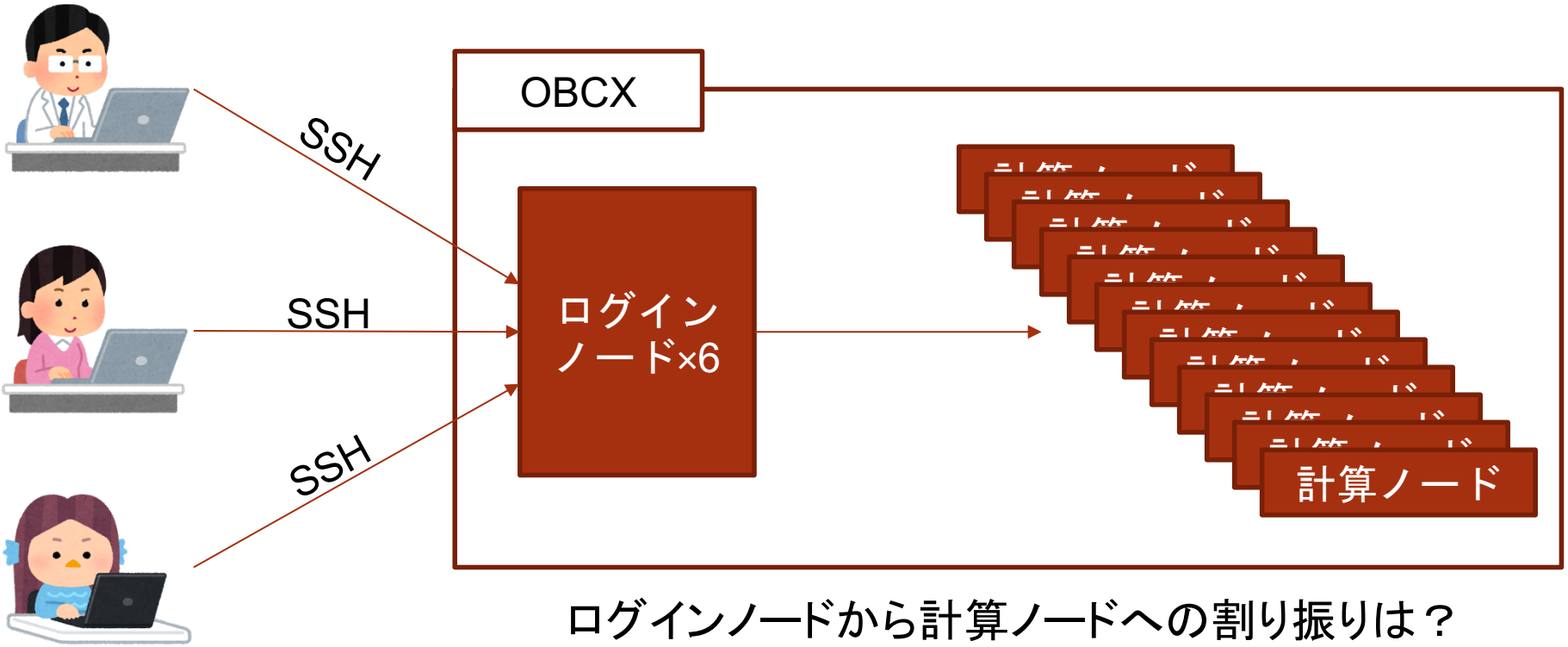
システムへのログイン、および必要な計算資源の要求が必要



1. スパコンへのログイン
2. コンパイル
3. ジョブの投入
4. 結果の確認

ログインノードとは？

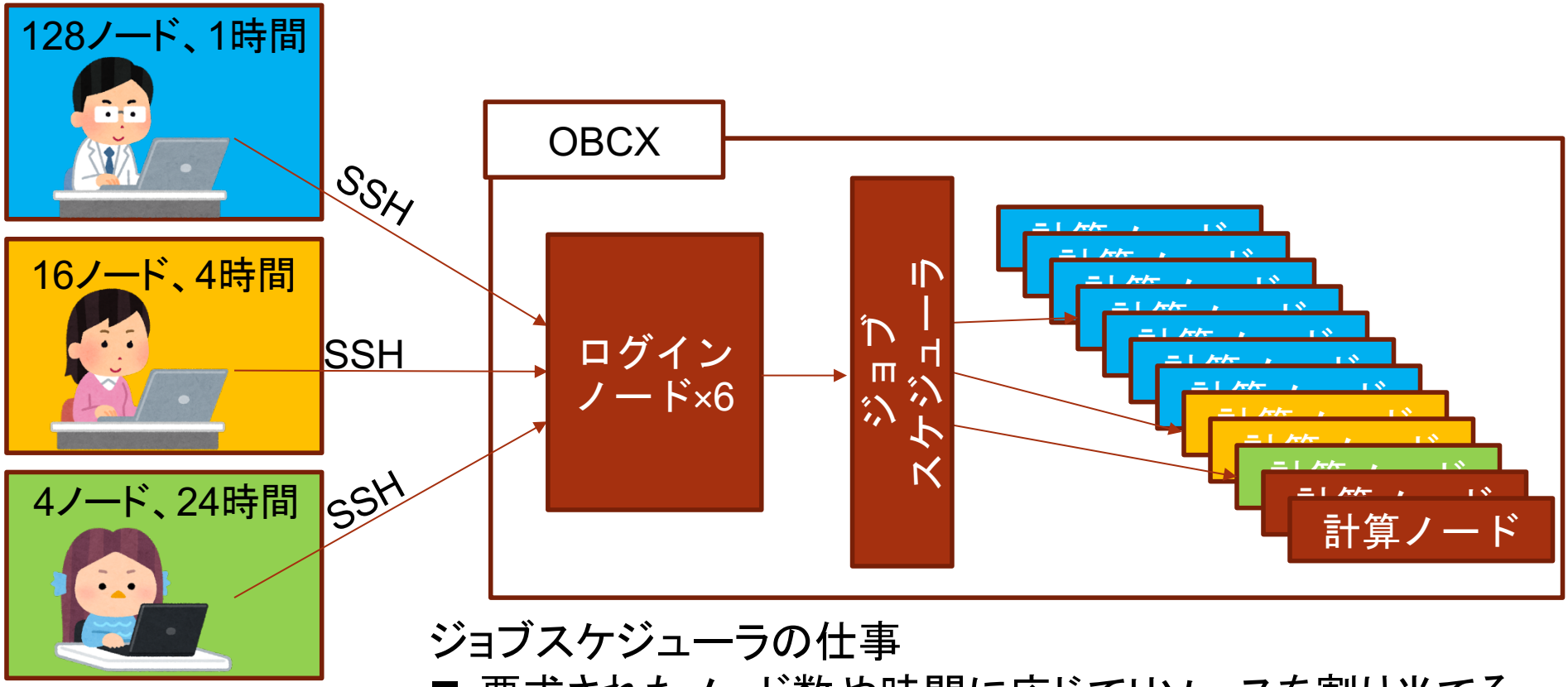
多くのユーザーで1つのシステムを使うためのノード
SSH（Secure Shell)プロトコルを使用してログイン



ログインノードから計算ノードへの割り振りは？

ジョブスケジューラ

ユーザーが要求するリソースを確保し、計算ノードを割り当てる機構

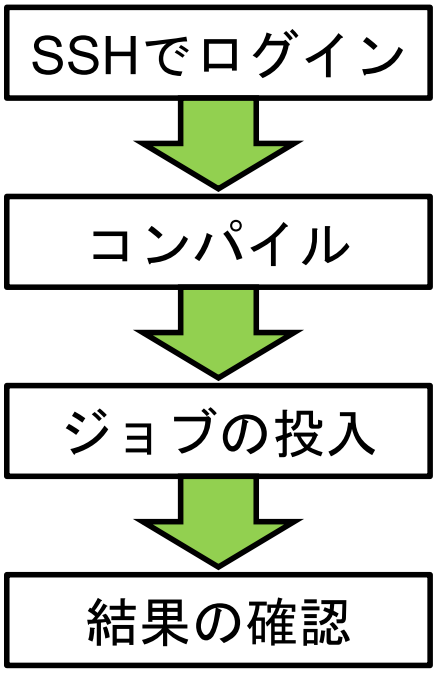


ジョブスケジューラの仕事

- 要求されたノード数や時間に応じてリソースを割り当てる。
  - すぐに実行されるわけではない。
  - ジョブを投入しておけば自動的に実行してくれる。
- エラーがあるとメールでユーザーに知らせる。

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



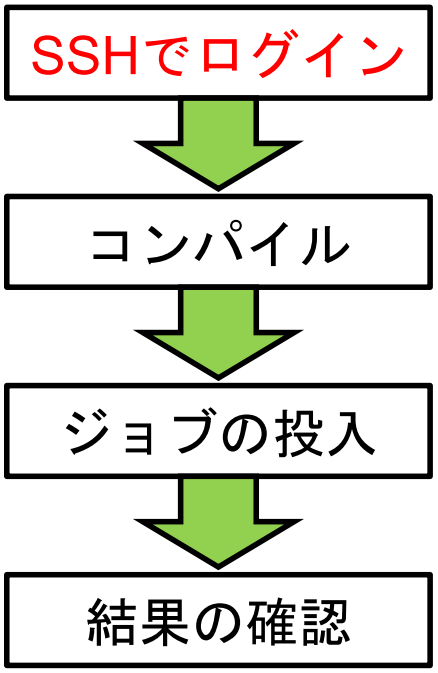
Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi, nano)、module

ジョブファイル、ジョブ管理

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi, nano)、module

ジョブファイル、ジョブ管理

Secure Shell プロトコル

通信が暗号化されたShell

- ShellはOSとユーザーの仲介をするコマンドベースのソフトウェア
- 通信データを暗号化し、リモートマシンにアクセスできる通信方法:SSH

暗号化された通信を使用して、様々なことが可能

- ファイルのコピー
- グラフィカル画面の転送
- トンネリング
- ディレクトリのマウント

ログイン後の画面の一例

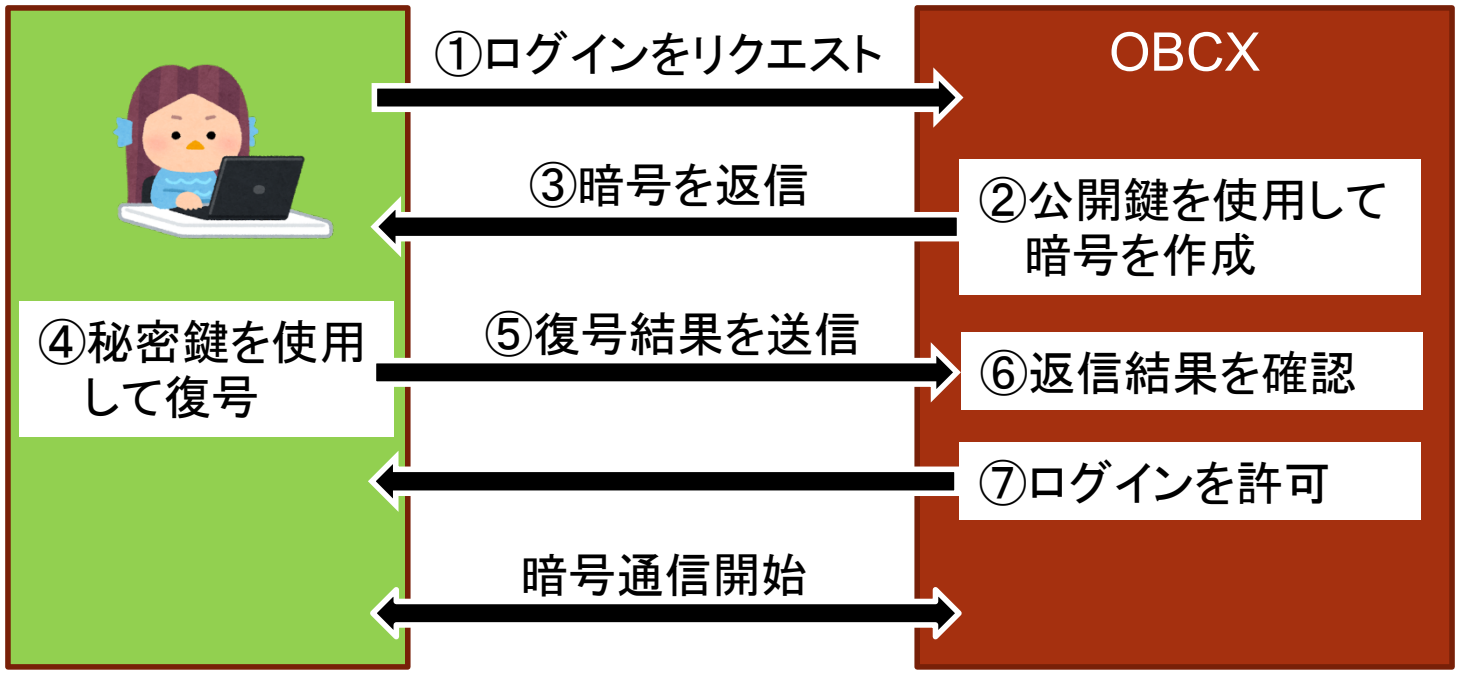
```
[ tUVXYZ @obcx05 ~]$ pwd
/home/ tUVXYZ
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/z30113
[ tUVXYZ @obcx05  tUVXYZ ]$ cd ../
[ tUVXYZ @obcx05 gt00]$ pwd
/work/gt00
[ tUVXYZ @obcx05 gt00]$ cd ~/
[ tUVXYZ @obcx05 ~]$ pwd
/home/z30113
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/z30113
[ tUVXYZ @obcx05  tUVXYZ ]$ mkdir test
[ tUVXYZ @obcx05  tUVXYZ ]$ ls
test
[ tUVXYZ @obcx05  tUVXYZ ]$
```

鍵認証方式

より安全な接続をする→鍵認証方式

- パスワードではなく、鍵ペアを使用してログインする方法
- 秘密鍵にもパスワードを設定可能

初期設定（初回ログイン時のみ）
- 鍵ペアを作成
- 公開鍵をログインノードに登録



鍵認証方式の注意点

注意点

■ 秘密鍵の取り扱いに注意
- 厳重に管理してください。
  - ✓ 漏洩すると容易にログインできてしまうため。
- 秘密鍵の入ったPCの紛失などがあった場合は速やかに公開鍵を更新してください。

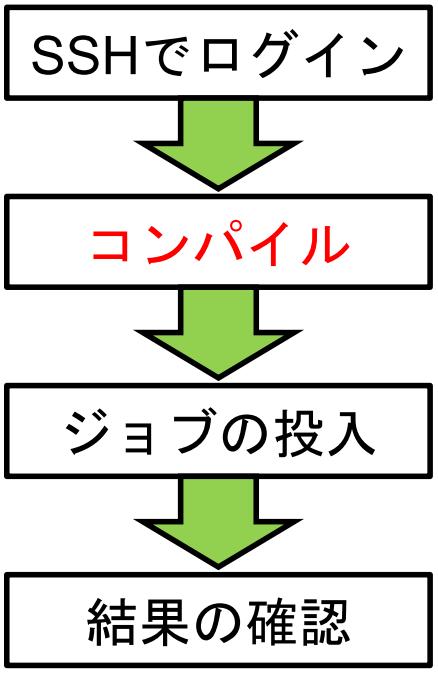
■ 鍵の生成時には必ずパスワードを設定してください。

よく間違える点

■ 秘密鍵のパスワードはOBCXのポータルのログインパスワードやログイン後のアカウントのパスワードとは異なります。

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi)、module

ジョブファイル、ジョブ管理

Linux OS

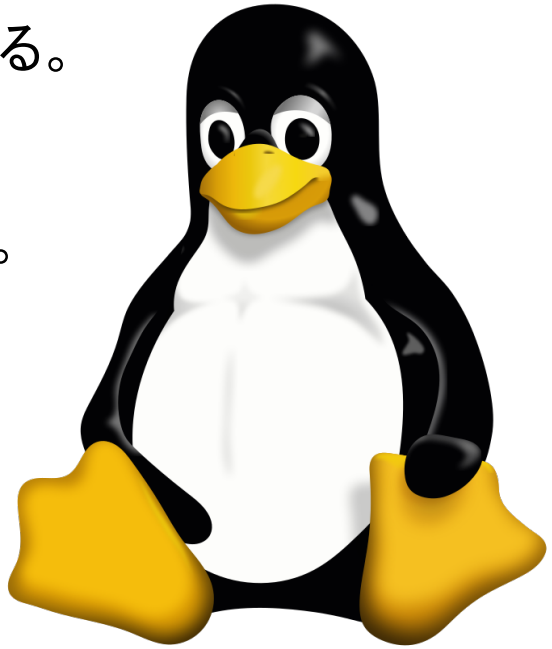
世界中のスパコンのほとんどが採用しているOS

- 現在のTOP500（上位500台のスパコンランキング）のシェアは100%
- スパコンだけでなく、サーバーの多くはLinux
- 基本的にOpenSourceで構成されている。
- CLI（Command Line Interface)で完結するよう、
  高機能なShellが用意されている。
  - 本講習ではBashを扱う
  - GUI(Graphical User Interface)も用意されている。
    スパコンではあまり使わない。

講習会では基本的なコマンドとBashについて扱います。

Linuxではフォルダをディレクトリといいます。
（この後頻出します。）

Tux



コマンドとは？

特定の機能を実行する命令

- コマンドの基本的なフォーマット
  $コマンド [オプション1、オプション2・・・] 引数1、引数2・・・・・
- オプションは“-”をつける。たとえば、
  $コマンド –h
  とすると、そのコマンドのヘルプが出力される。

基本的なコマンド一覧

コマンド	用途	よく使うオプション
pwd	現在のディレクトリを表示	
cd	ディレクトリを変更	
ls	ファイル一覧を表示	-l ： 詳細表示
cp	ファイルまたはディレクトリのコピー	-r ： 再帰的にコピー
mv	ファイル、またはディレクトリの移動	-r ： 再帰的に移動
rm	ファイルまたはディレクトリの削除	-r ： 再帰的に削除
mkdir	ディレクトリの作成	-r ： 最適的に作成
man	コマンドの説明	
exit	セッションの終了	

Bashの機能

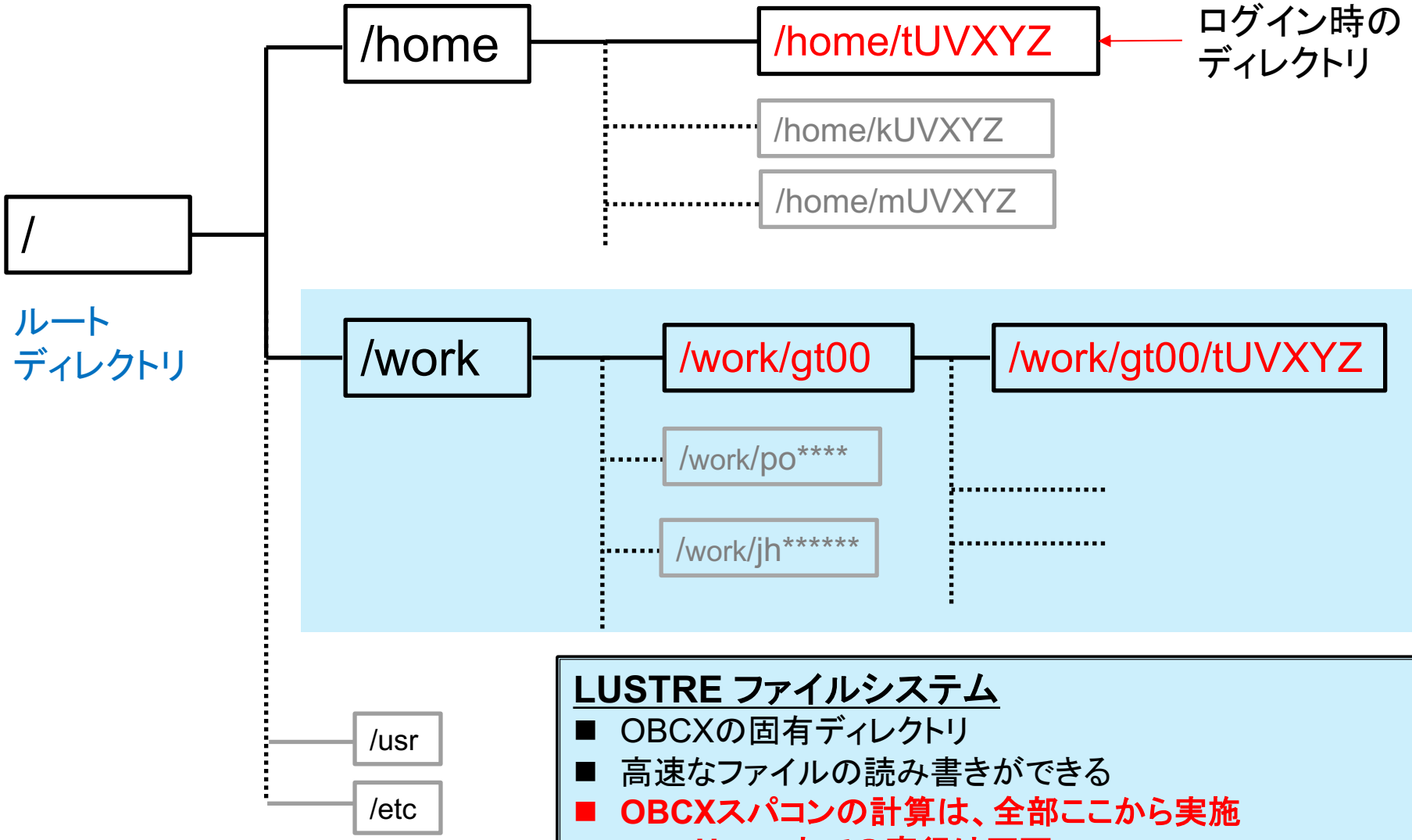
コマンド入力を簡易にする機能を提供

基本的な機能一覧

	操作	動作内容
Auto fill	tab	途中まで入力したコマンドやパスの残りを補完
履歴	↑	過去に実行したコマンドを1つずつさかのぼる
履歴探索モード	ctrl+r	過去に実行したコマンドを検索
リダイレクト	>	結果をファイルなどに出力 例：ls > list.txt　list.txtにlsの結果が出力される
パイプ	|	結果を次のコマンドに渡す 例：ls | sort　　lsコマンドの結果をソートする

コマンドとbashの機能を使用したスクリプトも記述可能

Linux(OBCX)でのディレクトリ構造



LUSTRE ファイルシステム

- OBCXの固有ディレクトリ
- 高速なファイルの読み書きができる
- **OBCXスパコンの計算は、全部ここから実施**
  - **Home上での実行は不可**
- トラブルもたまにあるので /homeから隔離されています

Linuxでのディレクトリ

Linuxでのディレクトリの参照方法

- ホームディレクトリ
  - “~/”
  - /home/tUVXYZと同じ
- 絶対パス
  - ルートディレクトリ“/“から始まるパス
    例：/home/tUVXYZ
- 相対パス
  - 現在のディレクトリから見たパス
  - “../”が1つ上のディレクトリ
    例： ../../work/
    現在のディレクトリが“home/tUVXYZ”とすると “/work”と同じ

コマンドの実行例

コマンドの実行例

```
[ tUVXYZ @obcx05 ~]$ pwd
/home/ tUVXYZ
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/ tUVXYZ
[ tUVXYZ @obcx05  tUVXYZ ]$ cd ../
[ tUVXYZ @obcx05 gt00]$ pwd
/work/gt00
[ tUVXYZ @obcx05 gt00]$ cd ~/
[ tUVXYZ @obcx05 ~]$ pwd
/home/ tUVXYZ
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/ tUVXYZ
[ tUVXYZ @obcx05  tUVXYZ ]$ mkdir test
[ tUVXYZ @obcx05  tUVXYZ ]$ ls
test
[ tUVXYZ @obcx05  tUVXYZ ]$
```

現在のディレクトリを出力

講習会用ディレクトリに移動

1つ上のディレクトリに移動

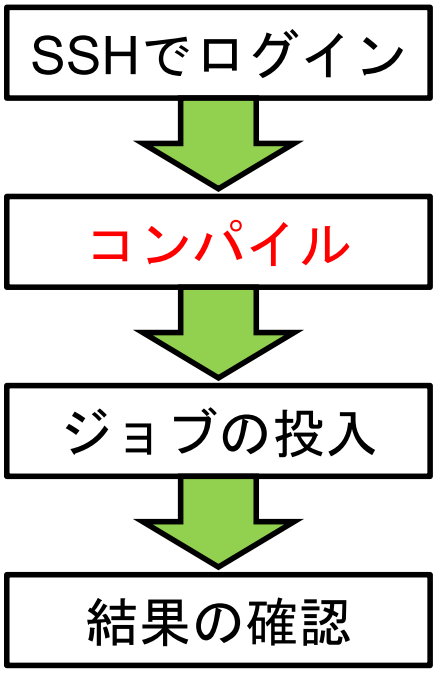
ホームディレクトリに移動

testディレクトリを作成

現在のディレクトリにあるファイルおよびディレクトリー覧を表示

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi, nano)、module

ジョブファイル、ジョブ管理

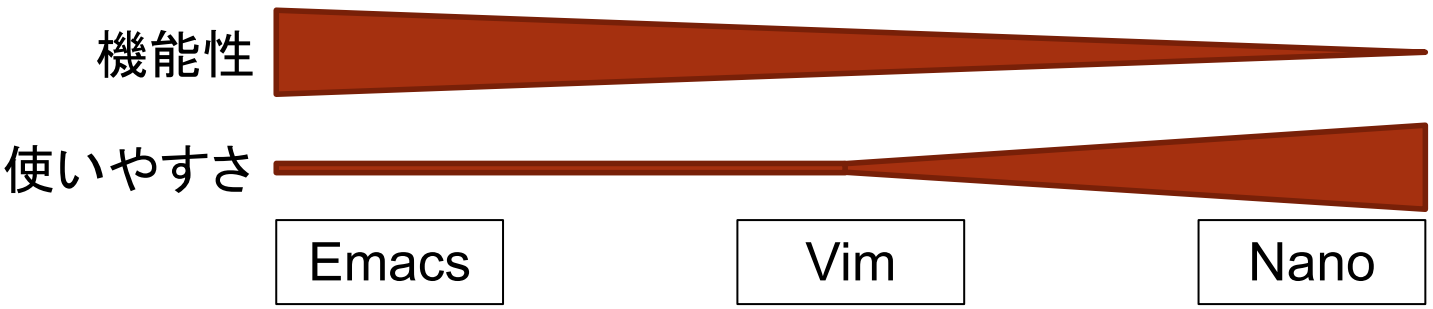
エディタ

ファイルを編集するためのソフトウェア

Windowsの標準だとメモ帳に該当 → ただし、もっと高機能かつマウスなしで操作可

Linuxでは以下のエディタが有名（ほかにもたくさんあります）

- Emacs
  - 高機能
  - 拡張機能も豊富 (LISPで記述されている)
- Vim
  - 軽量
  - Linuxの初期状態でインストールされている
- Nano
  - 一番簡易
  - コマンドが下段に表示されるため、わかりやすい



Emacs

Emacsの基本的な操作方法について

編集画面

コマンド確認画面

$emacs で起動

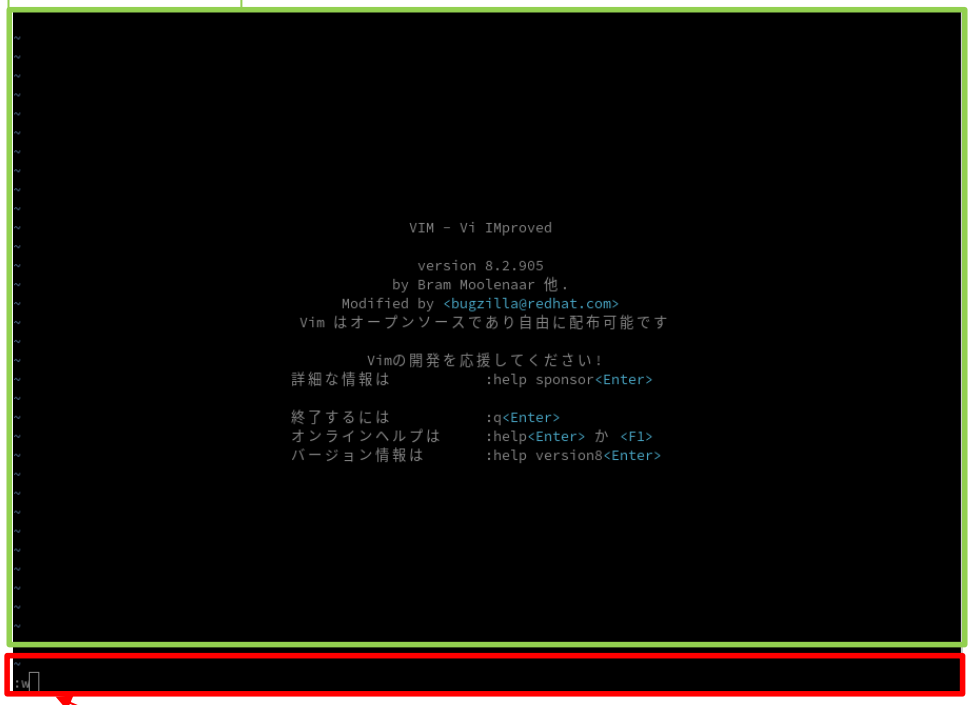
- C→Ctrlキー、M→Metaキー（ESC）、- → 同時押し
- Ctrl-zを押さないように注意

操作	コマンド
ファイルを開く	C-x, C-f ファイル名 Ret
別名保存	C-x, C-w ファイル名 Ret
上書き保存	C-x, C-s
Mark Set	C-space
コピー	Mark Set, 範囲選択, M, w
切り取り	Mark Set, 範囲選択, C-w
貼り付け	C-y
検索	C-s
Undo	C-x, u
キャンセル	C-g
終了	C-x, C-c

Vim

Vimの基本的な操作方法について

編集画面



モードおよびコマンド確認画面
- ブランク → ノーマルモード
- : → コマンドモード
  - :w Ret 上書き保存
- --挿入-- → 入力モード

$vim で起動

Vimではノーマルモード、入力モードとコマンドラインモードを切り替えて使う

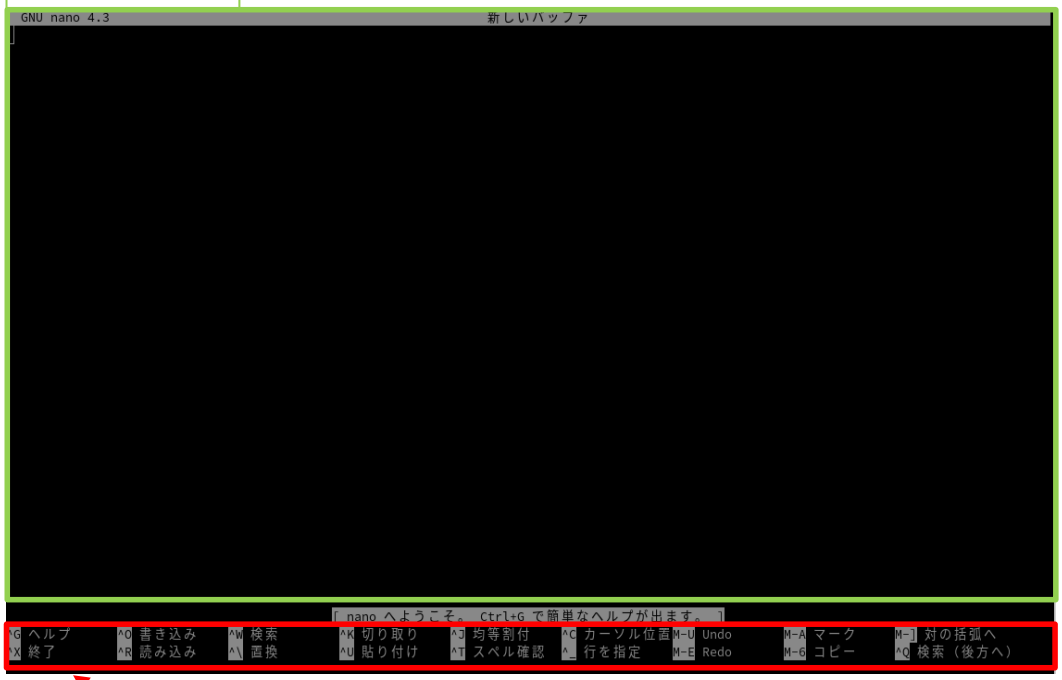
現在のモード	移行先モード	キー
ノーマル	入力	i, a, Ins
ノーマル	コマンドライン	:
入力	ノーマル	esc, ctrl-c

操作	コマンド
ファイルを開く	e Ret
別名保存	w ファイル名 Ret
上書き保存	w Ret
1行コピー	yy
1行切り取り	dd
貼り付け	p
検索	/*検索対象* Ret
終了	q

Nano

Nanoの基本的な操作方法について

編集画面



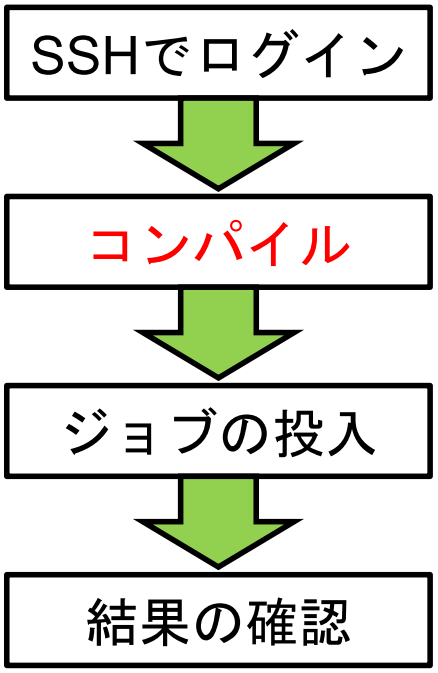
コマンド確認画面
- ^ → Ctrl
- M → Esc
- - → 順次押す

$nano で起動
- ^G → CtrlとGキーを同時押し
- M-U → Escキー、Uキーを順番に押す

操作	コマンド
ファイルを開く	^R
別名保存	^O ファイル名 Ret
上書き保存	^O
マーク	M-A
マークキャンセル	M-A
コピー	M-6
切り取り	^K
貼り付け	^U
検索	^W
終了	^X

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi, nano)、module

ジョブファイル、ジョブ管理

コンパイル、実行環境構築

コンパイル、実行環境を整えるために、moduleコマンドを使用

$ module [オプション] *引数*

オプション	内容
avail	利用可能な環境の一覧を表示
list	現在ロードしている環境一覧を表示
load	指定した環境のロード
unload	指定した環境のアンロード
switch	環境のロードとアンロードを同時に実行
purge	環境を全てアンロード

例えば、Pythonを使用する場合、

$ module load python/3.7.3

プログラムのコンパイル

IntelおよびGNUコンパイラがインストール済み

ソースコードのタイプ	コンパイラ開発元	言語	呼び出しコマンド
非MPIコード	Intel	C	icc
		Fortran	ifort
	GNU	C	gcc
		Fortran	gfortran
MPIコード	Intel	C	mpiicc
		Fortran	mpiifort
	GNU	C	mpicc
		Fortran	mpif77 or mpif90

IntelでC言語のコードをコンパイルする場合
$ icc [オプション] “ソースコード.c” –o “出力ファイル名”
GNUでFortran90/95のコードをコンパイルする場合
$ gfortran [オプション] -free-line-length-none “ソースコード.f90” –o “出力ファイル名”

-free-line-length-none : Fortranの自由形式を有効化

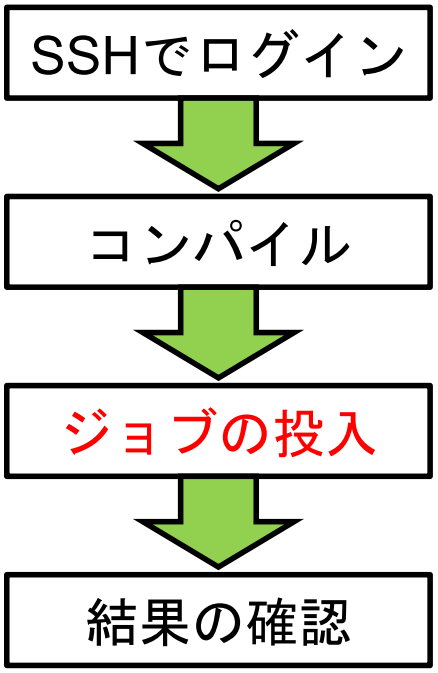
コンパイルオプション一例

IntelおよびGNUでよく使うコンパイルオプション一覧

タイプ	言語	Intel オプション	GNU オプション	内容
共通	共通	-qopenmp	-fopenmp	OpenMPを有効化
デバッグ オプション	共通	-g		実行ファイルにソースコードの情報を付与
		-Wall		コンパイル時にバグの元になりそうな箇所を検知
		-traceback	-fbacktrace	実行時にエラー発生個所を特定
	Fortran	-check bounds	-fbounds-check	実行中に初期化していない値へのアクセスなどを検知
最適化 オプション	共通	-O0、-O1、-O2、-O3		数字が大きいほど積極的な最適化を適用
		-xHost	-march=native	CPUアーキテクチャを考慮した最適化を有効化

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



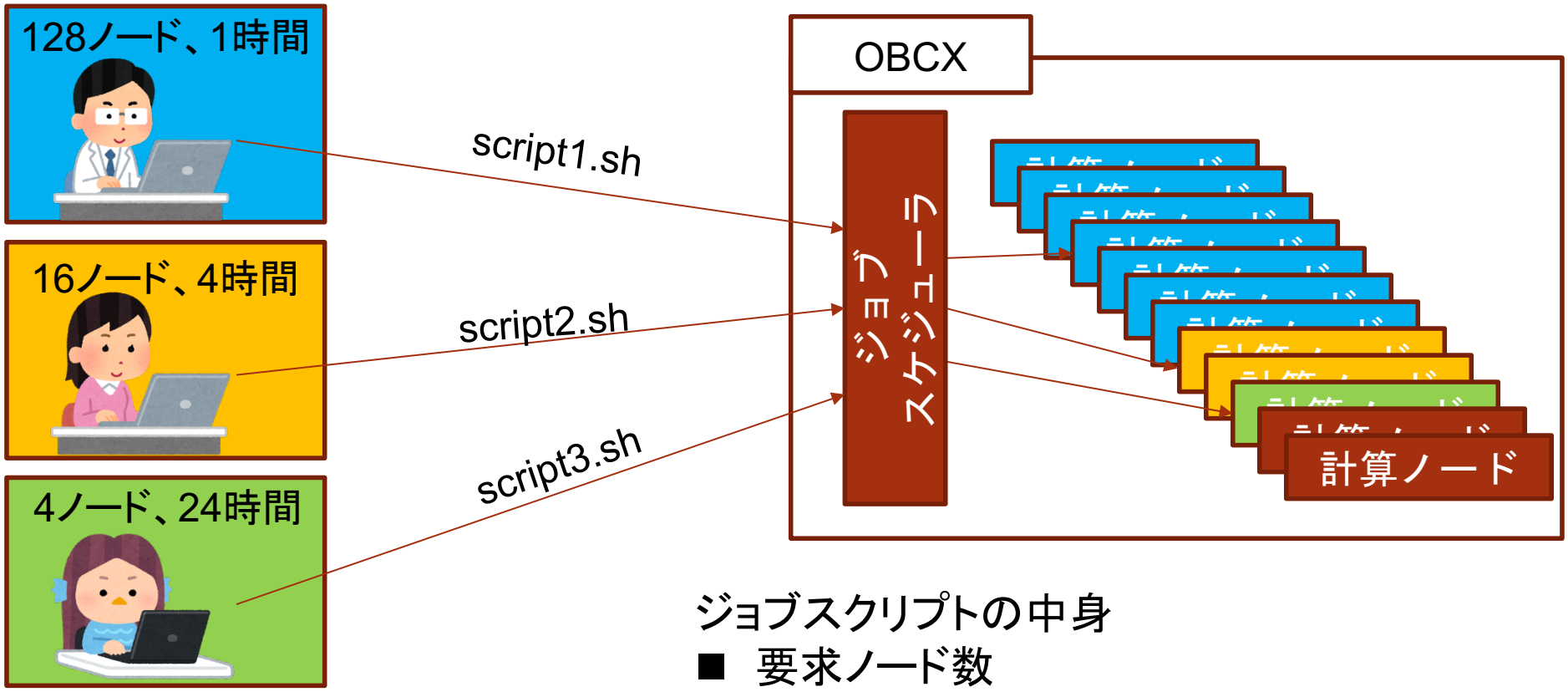
Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi, nano)、module

ジョブファイル、ジョブ管理

ジョブスケジューラを使うためには

ジョブスクリプトを用意し、コマンドでジョブを投入



ジョブスクリプトの中身

- 要求ノード数
- 予想実行時間
- ジョブグループ
- 実行コマンド　　など・・・・・

ジョブスクリプトの作成

ジョブスクリプトはジョブスケジューラーへのオプション群を最初に記述

サンプルジョブスクリプト

```
#!/bin/bash
#PJM -L rscgrp=lecture    #リソースグループの指定(後述)
#PJM -L node=4            #要求ノード数
#PJM --mpi proc=8         #MPIプロセス数
#PJM --omp thread=28      #1MPIプロセス辺りのスレッド数
#PJM -L elapse=00:15:00   #ジョブの推定実行時間
#PJM -g gt00              #ジョブグループの指定

#------- Program execution -------#
mpiexec.hydra -n ${PJM_MPI_PROC} ./a.out  #プログラム実行
 .
 .
```

ジョブスケジューラに渡されるオプション群

通常のシェルスクリプトとしても使用可能

リソースグループ

リソースグループ毎のノード数および最長実行時間一覧

リソースグループ名	ノード数	最長実行時間	説明
debug	1～16	30分	デバッグ用
short	1～8	8時間	短いジョブ用
regular	1～128	48時間	通常使うリソースグループ
	129～256	24時間	
interactive	1	2時間	インタラクティブジョブ用（後述）
	2～8	10分	
tutorial	1～8	15分	講習会用
lecture	1～8	15分	講習会後一か月間有効

- 講習会中はtutorialをご使用ください。
- 講習会後から利用期限まではlectureをご利用ください。
- それ以外のリソースグループは使用できません。

ジョブの管理コマンド

以下のコマンドを使用して、ジョブの投入、削除、確認を行う

コマンド	内容	使い方
pjsub	ジョブの投入	$pjsub “script.sh”
pjdel	ジョブの削除	$pjdel “job ID”
pjstat	ジョブの状態	$pjstat

注＊これらのコマンドはスパコンによって異なる

pjstatのオプション

- -H　　：終了したジョブの確認
- --rsc -b：各リソースグループの混雑具合を確認可能
- --rsc -x：各リソースグループで要求可能なリソース量を出力

インタラクティブジョブ

対話型のジョブ デバッグなどに便利

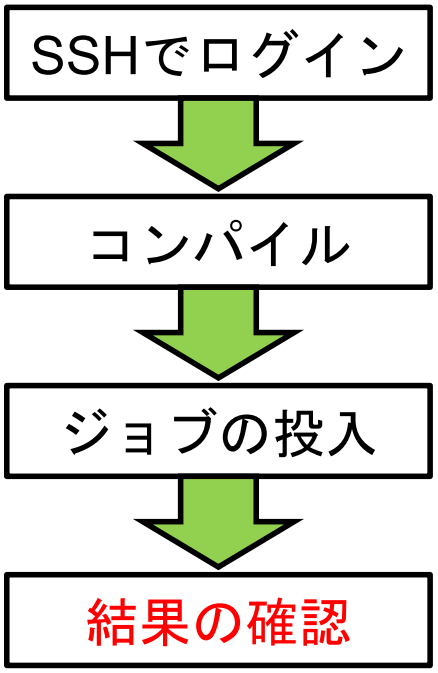
手元で実行しているのと同様の挙動

```
[tUVXYZ@obcx04 tUVXYZ]$ pjsub --interact -g gt00 -L rscgrp=interactive,node=1
[INFO] PJM 0000 pjsub Job 517079 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 517079 started.
[tUVXYZ@cx0065 tUVXYZ]$
```

ログインノード(obcx04)から計算ノード(cx0065)へ
手元で操作するのと同様にプログラムの実行が可能

スパコンを使うための手順 (本日の実習内容)

本日の実習内容はスパコンへのログインからプログラムの実行、結果の確認まで



Putty、Cygwin、SSH、鍵認証

Linuxコマンド、エディタ(emacs, vi)、module

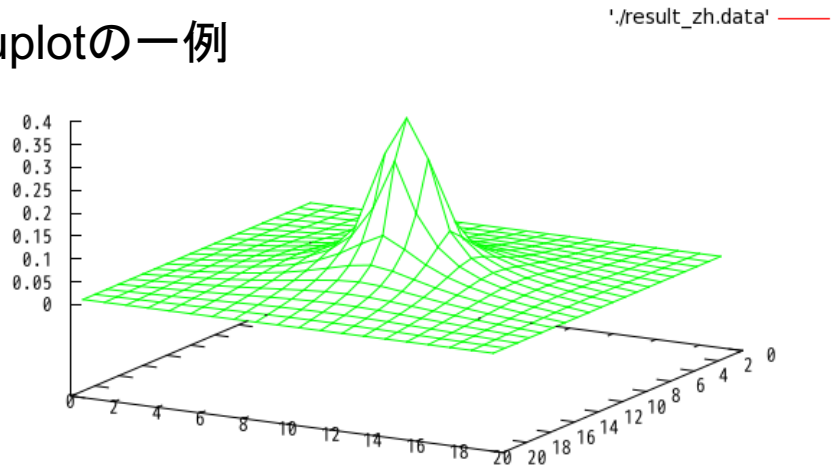
ジョブファイル、ジョブ管理

結果の確認

Linux上でデータを整理するかローカルマシンにコピー

- Linux上
  - コマンドを駆使してデータを成型（実習内容）
  - gnuplotなどのグラフ作成ソフトを使用
- ローカルにコピー
  - SFTPプロトコルを使用してデータを手元にコピー
    実習ではWinSCPまたはsftpコマンドを使用

gnuplotの一例



以上が講義の内容になります。
休憩の後、実習になります。

ご質問等はSlackに書き込み、または
Zoomの挙手にてお願いします。

利用可能なライブラリー覧

```
---------------------------- /home/opt/local/modulefiles/L/mpi/intel/2019.5.281/impi/2019.5.281 ----------------------------
alps/2.3.0(default)                  openmx/3.8(default)                  ppohFDM/0.3.1(default)
feram/0.26.04(default)               parmetis/4.0.3(default)              ppohFEM/1.0.1(default)
frontflow_blue/8.1(default)          petsc/3.11.2(default)                ppohFVM/0.3.0(default)
frontistr/4.5(default)               phase/2019.01(default)               pt-scotch/6.0.6(default)
modylas/1.0.4(default)               phdf5/1.10.5(default)                revocap_coupler/2.1(default)
mpi-fftw/3.3.8(default)              pnetcdf/1.11.2(default)              superlu_dist/6.1.1(default)
netcdf-fortran-parallel/4.4.5(default) ppohBEM/0.5.0(default)             xtapp/rc-150401(default)
netcdf-parallel/4.7.0(default)       ppohDEM_util/1.0.0(default)

---------------------------------- /home/opt/local/modulefiles/L/compiler/intel/2019.5.281 ----------------------------------
R/3.6.0(default)              hdf5/1.10.5(default)          metis/5.1.0(default)          ppohAT/1.0.0(default)
akaikkr/cpa2002v010(default)  hdf5/1.8.21                  mt-metis/0.6.0(default)       revocap_refiner/1.1.04(default)
bioconductor/3.10(default)    impi/2019.5.281(default)     netcdf/4.7.0(default)         samtools/1.9(default)
blast/2.9.0(default)          intelpython/2.7              netcdf-cxx/4.3.0(default)     scotch/6.0.7(default)
bwa/0.7.17(default)           intelpython/3.6(default)     netcdf-fortran/4.4.5(default) superlu/5.2.1(default)
fftw/3.3.8(default)           mesa/19.0.6(default)         paraview/5.6.1(default)       superlu_mt/3.1(default)
gsl/2.5(default)              metis/4.0.3                  povray/3.7.0.8(default)       xabclib/1.03(default)

------------------------------------------- /home/opt/local/modulefiles/L/core -------------------------------------------
acusolve/2019.1.0(default)       gcc/7.5.0                    itac/2019.5.041(default)
advisor/2019.3.0.591490          go/1.12.6(default)           julia/1.4.0(default)
advisor/2019.4.0.597843          hyperworks/2019.1.0(default) llvm/7.1.0(default)
advisor/2019.5.0.602216(default) inspector/2019.3.0.591484    massivethreads/0.97
anaconda/2-2019.03               inspector/2019.4.0.597413    massivethreads/0.99(default)
anaconda/3-2019.03(default)      inspector/2019.5.0.602103(default) python/2.7.16
bioperl/1.007002(default)        intel/2017.4.196             python/3.7.3(default)
bioruby/1.5.2(default)           intel/2018.3.222             singularity/3.2.1(default)
cmake/3.0.2                      intel/2019.3.199             texlive/2020(default)
cmake/3.14.5(default)            intel/2019.4.243             vtune/2019.3.0.591499
devtoolset/7(default)            intel/2019.5.281(default)    vtune/2019.4.0.597835
gaussian/16.C.01(default)        intel/2020.1.217             vtune/2019.6.0.602217(default)
gcc/4.8.5(default)               itac/2019.4.036              xcrypt/1e8a958966aa(default)
```

```
[ tUVXYZ @obcx05 ~]$ pwd
/home/ tUVXYZ
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/ tUVXYZ
[ tUVXYZ @obcx05  tUVXYZ ]$ cd ../
[ tUVXYZ @obcx05 gt00]$ pwd
/work/gt00
[ tUVXYZ @obcx05 gt00]$ cd ~/
[ tUVXYZ @obcx05 ~]$ pwd
/home/ tUVXYZ
[ tUVXYZ @obcx05 ~]$ cd /work/gt00/ tUVXYZ
[ tUVXYZ @obcx05  tUVXYZ ]$ mkdir test
[ tUVXYZ @obcx05  tUVXYZ ]$ ls
test
[ tUVXYZ @obcx05  tUVXYZ ]$
```