内容に関するご質問は
ida@cc.u-tokyo.ac.jp
まで、お願いします。

[Reedbush編]

第120回　お試しアカウント付き
並列プログラミング講習会
「 Altair HyperWorks実行」

スパコンへのログイン・
テストプログラム起動

東京大学情報基盤センター　特任准教授　伊田 明弘

スパコンへのログイン・ファイル転送・基本コマンド

Reedbushへログイン

- ターミナルから、以下を入力する

  $ ssh reedbush-u.cc.u-tokyo.ac.jp -l tYYxxx

  「-l」はハイフンと小文字のL、

  「tYYxxx」は利用者番号（数字）

  “tYYxxx”は、利用者番号を入れる
- 接続するかと聞かれるので、 yes を入れる
- 鍵の設定時に入れた

  自分が決めたパスワード（パスフレーズ）

  を入れる
- 成功すると、ログインができる

Reedbushにおける注意

- ログインするとホームディレクトリ(/home/gt00/t001XX)にいます。
- /home ファイルシステムは容量が小さく、ログインに必要なファイルだけを置くための場所です。
  - /home に置いたファイルは計算ノードから参照できません。ジョブの実行もできません。
- 計算に必要なファイルは、/lustre ファイルシステムに移動(mv)させてください。

- ホームディレクトリ: /home/gt00/t001XX
  - cd コマンドで移動できます。
- Lustreディレクトリ: /lustre/gt00/t001XX
  - cdw コマンドで移動できます。

PCのファイルをReedbushに置く

- ターミナルから、以下を入力する

$ scp ./a.f90 tYYxxx@reedbush-u.cc.u-tokyo.ac.jp:

「tYYxxx」は利用者番号（数字）

“tYYxxx”は、利用者番号を入れる

- PCのカレントディレクトリにある”a.f90”を、Reedbush上のホームディレクトリに置く
- ディレクトリごと置くには、”-r” を指定

$ scp -r ./SAMP tYYxxx@reedbush-u.cc.u-tokyo.ac.jp:

- PCのカレントディレクトリにあるSAMPフォルダを、その中身ごと、Reedbush上のホームディレクトリに置く

Reedbushのデータを PCに取り込む

- ターミナルから、以下を入力する

$ scp tYYxxx@reedbush-u.cc.u-tokyo.ac.jp:~/a.f90 ./

「tYYxxx」は利用者番号（数字）
“tYYxxx”は、利用者番号を入れる

- Reedbush上のホームディレクトリにある”a.f90”を、PCのカレントディレクトリに取ってくる
- ディレクトリごと取ってくるには、”-r” を指定

$ scp -r tYYxxx@reedbush-u.cc.u-tokyo.ac.jp:~/SAMP ./

- Reedbush上のホームディレクトリにあるSAMPフォルダを、
その中身ごと、PCのカレントディレクトリに取ってくる

ＵＮＩＸ備忘録

- emacsの起動： emacs　編集ファイル名
  - ^x　^s　（^はcontrol）：テキストの保存
  - ^x　^c　：　終了
    （ ^z で終了すると、スパコンの負荷が上がる。絶対にしないこと。）
  - ^g　：　訳がわからなくなったとき。
  - ^k　：カーソルより行末まで消す。
    消した行は、一時的に記憶される。
  - ^y　：^kで消した行を、現在のカーソルの場所にコピーする。
  - ^s　文字列：　文字列の箇所まで移動する。
  - ^M x　goto-line：　指定した行まで移動する。

ＵＮＩＸ備忘録

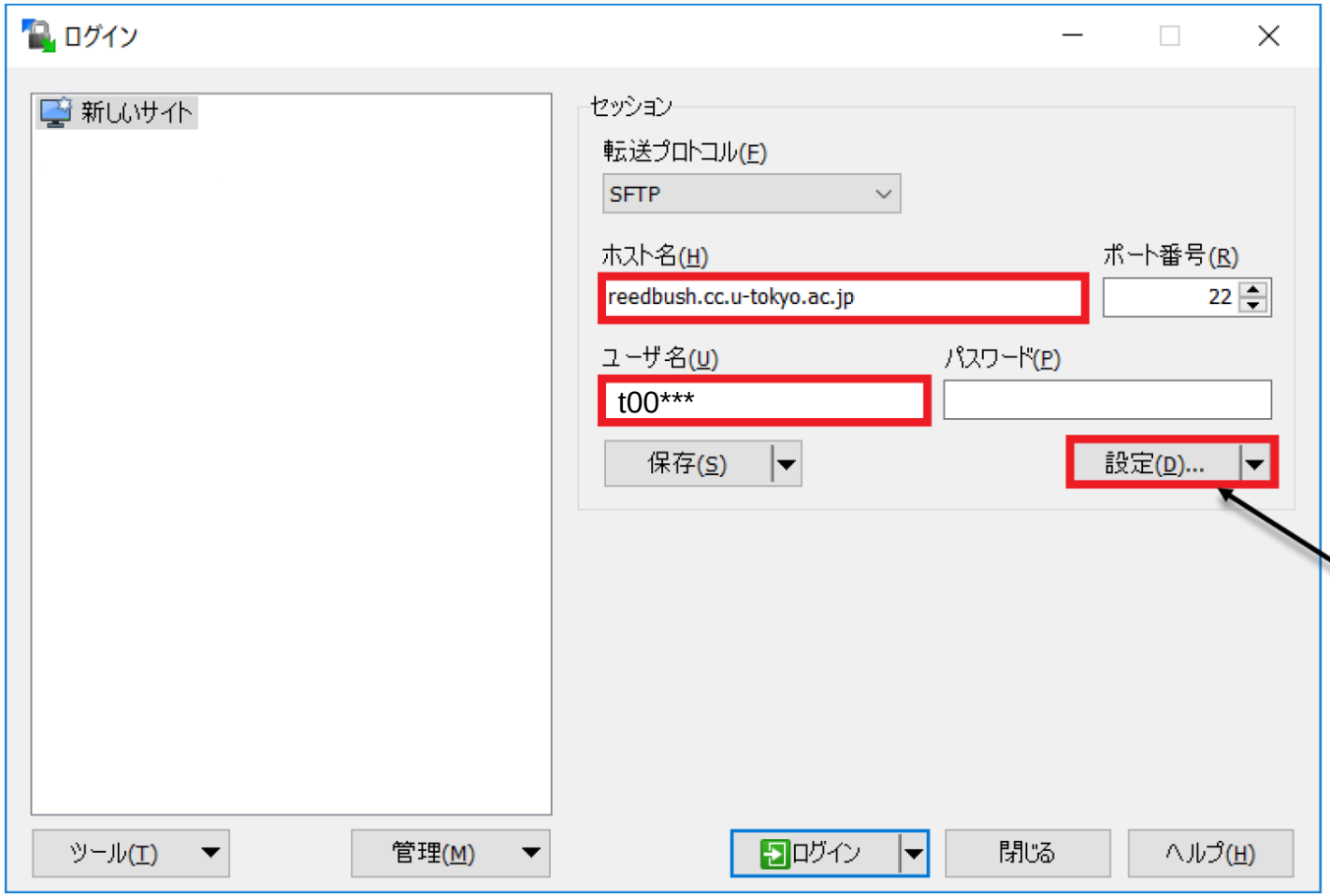
- rm ファイル名： ファイル名のファイルを消す。
  - rm *~： test.c~ などの、~がついたバックアップファイルを消す。使う時は慎重に。*~ の間に空白が入ってしまうと、全てが消えます。
- ls： 現在いるフォルダの中身を見る。
- cd フォルダ名： フォルダに移動する。
  - cd .. ：一つ上のフォルダに移動。
  - cd ~ ：ホームディレクトリに行く。訳がわからなくなったとき。
- cat ファイル名： ファイル名の中身を見る
- make：実行ファイルを作る
  （Makefile があるところでしか実行できない）
  - make clean：実行ファイルを消す。
    （clean がMakefileで定義されていないと実行できない）

ＵＮＩＸ備忘録

- less　ファイル名：　ファイル名の中身を見る(catでは画面がいっぱいになってしまうとき）
  - スペースキー：1画面スクロール
  - /　：文字列の箇所まで移動する。
  - q　：　終了（訳がわからなくなったとき）
- cp ファイル名　フォルダ名:ファイルをコピーする
- mv ファイル名　フォルダ名:ファイルを移動させる

WinSCPによるReedbushへのファイル転送

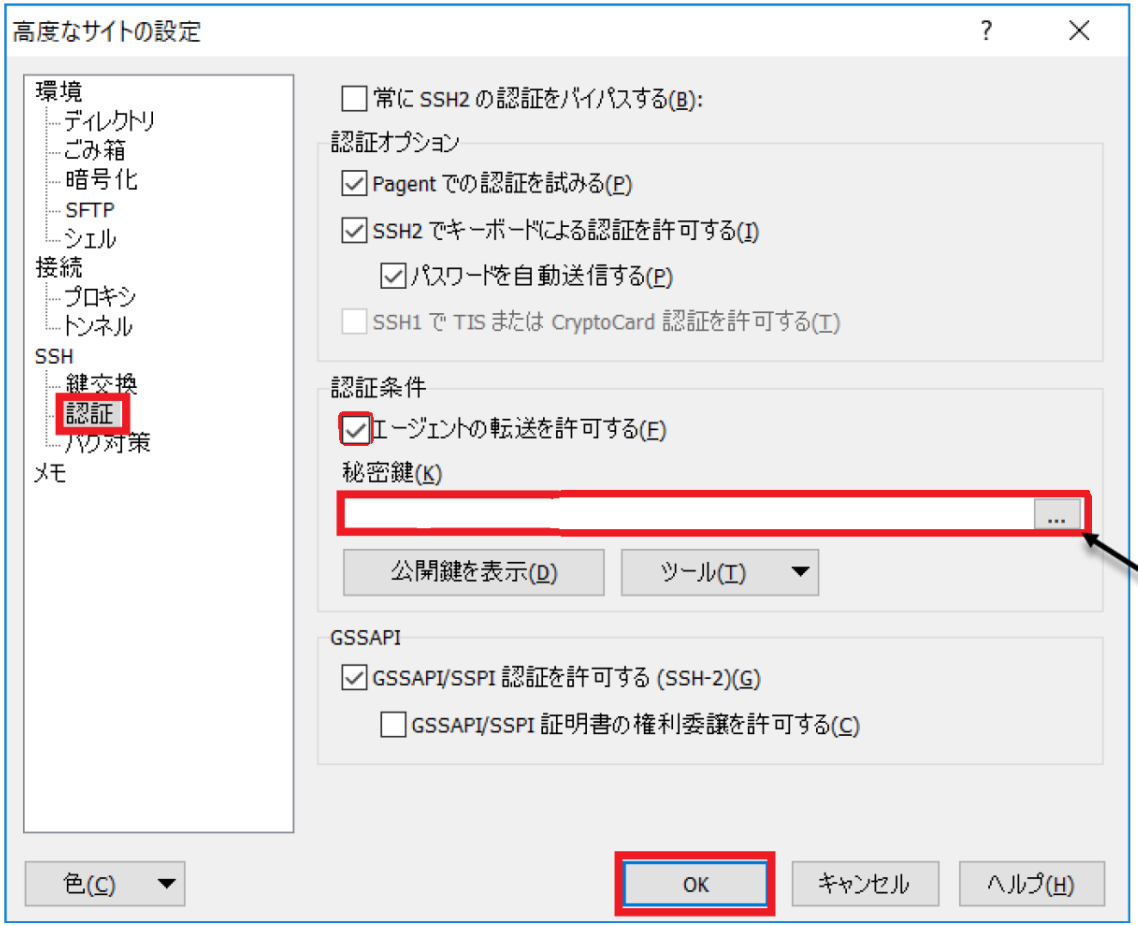
- Reedbushへの接続



クリックして
次頁へ

WinSCPによるReedbushへのファイル転送

- Reedbushへの接続の続き



クリックして
鍵ファイル選択

スパコン上でのプログラムの実行 [ReedBush編]

「ジョブ」の実行形態と実行方法について

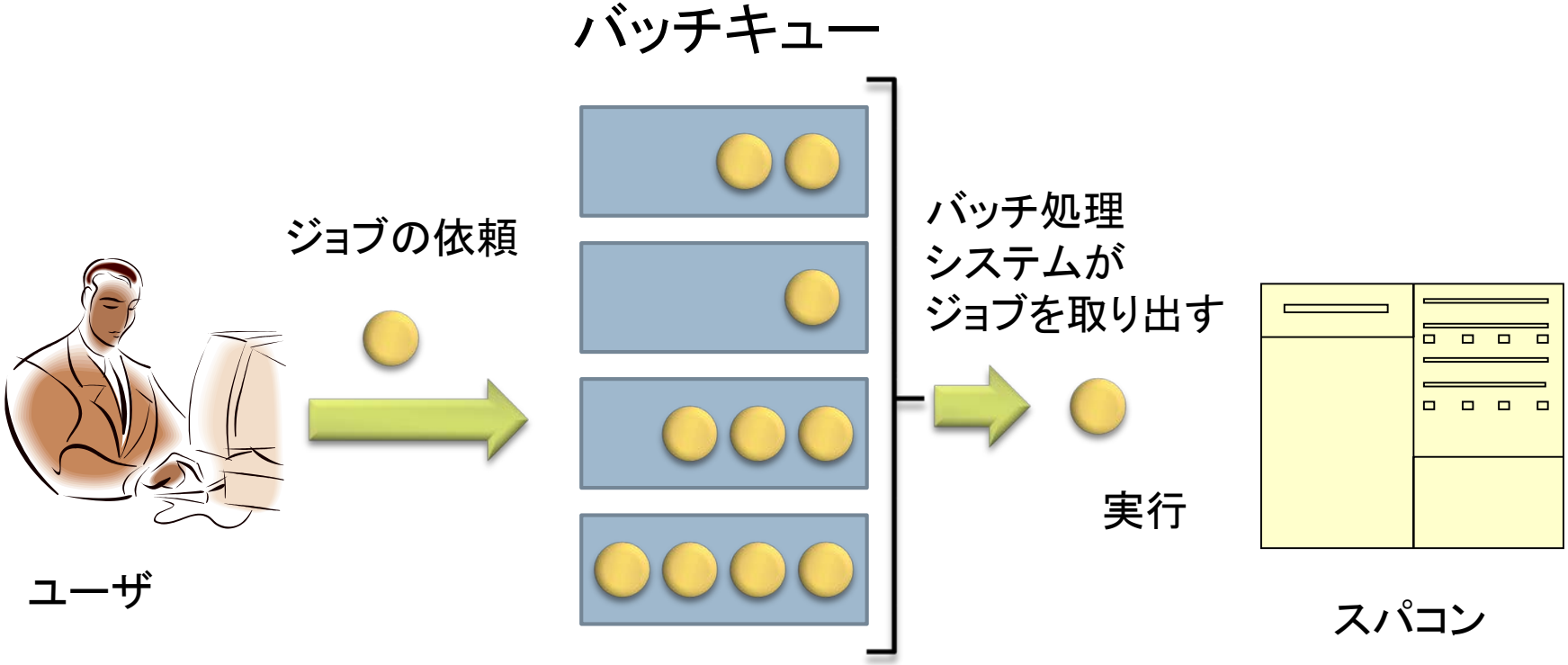
Reedbush-Hスーパーコンピュータシステムでのジョブ実行形態

- 以下の2通りがあります
- インタラクティブジョブ実行
  - PCでの実行のように、コマンドを入力して実行する方法
  - スパコン環境では、あまり一般的でない
  - デバック用、大規模実行はできない
  - Reedbush-Hでは、以下に限定
    - 1ノード（36コア, 2GPU）: 15分まで
    - 2ノード（72コア, 4GPU）: 5分まで
- バッチジョブ実行
  - バッチジョブシステムに処理を依頼して実行する方法
  - 実行させたい処理をファイル（ジョブスクリプト）で指示する
  - スパコン環境で一般的
  - 大規模実行用
  - Reedbush-Uでは、最大128ノード（4,608コア）(24時間)

※講習会アカウントではバッチジョブ実行のみ、最大2ノード10分まで

バッチ処理とは

- スパコン環境では、通常は、インタラクティブ実行（コマンドラインで実行すること）はできません。
- ジョブはバッチ処理で実行します。



バッチ処理を用いたジョブの実行方法

- Reedbushシステムにおいてバッチ処理は、Altair社のバッチシステム PBS Professionalで管理されています。
- ジョブの投入：
  qsub <ジョブスクリプトファイル名>

```
#!/bin/bash
#PBS -q h-lecture
#PBS -Wgroup_list=gt00h
#PBS -l select=1
#PBS -l walltime=00:01:00
cd $PBS_O_WORKDIR
. /etc/profile.d/modules.sh
./hello
```

キュー名
：h-lecture

利用グループ名
：gt00h

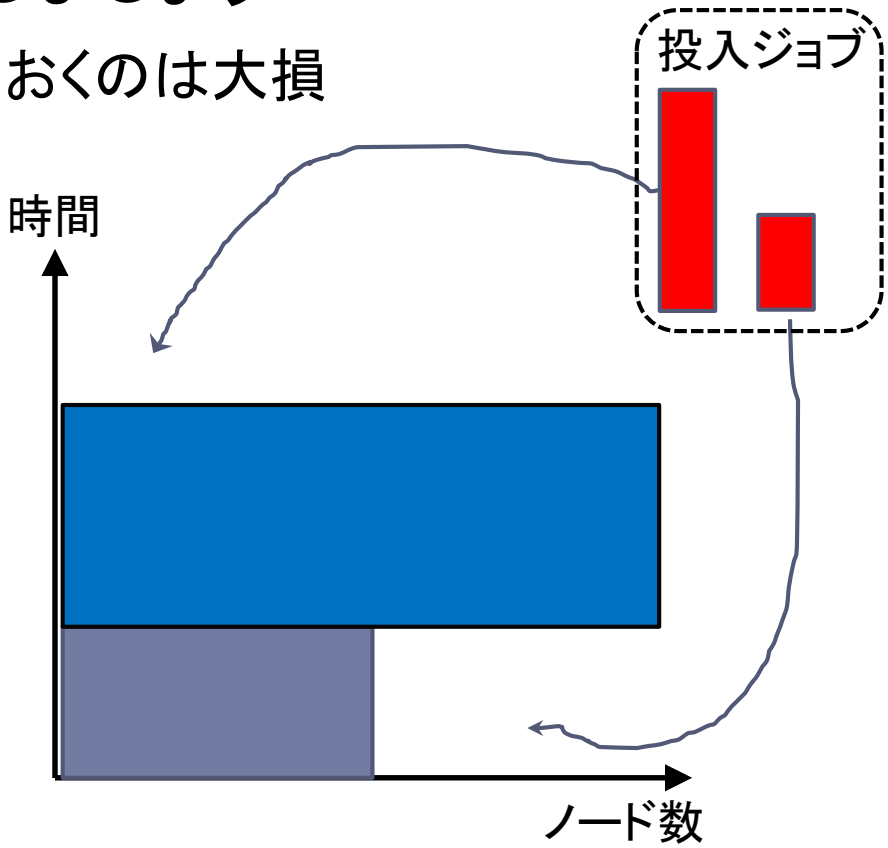
ジョブスクリプトファイルの例

ジョブ待ち時間を減らすポイント

- 「walltime」を適切に設定しましょう！
  - とりあえずと、最大時間を書いておくのは大損

```
#!/bin/bash
#PBS -q h-lecture
#PBS -Wgroup_list=gt00h
#PBS -l select=1
#PBS -l walltime=00:01:00
cd $PBS_O_WORKDIR
. /etc/profile.d/modules.sh
./hello
```

ジョブスクリプトファイルの例



Backfillのイメージ図

本講習会でのグループ名とキュー名

- グループ: gt00

  課金情報（財布）を管理するのに使用される

- キュー 名　: h-tutorial

  キューとは、スパコンにバッチジョブを
  投入する時の待ち行列の名前
  （詳細は後述）

本お試し講習会でのキュー・グループ名

- 本演習中のキュー名：
  - h-tutorial
  - 最大10分まで
  - 最大ノード数は2ノード(4GPU) まで
- 本演習時間以外（24時間）のキュー名：
  - h-lecture
  - 利用条件は演習中のキュー名と同様
- グループ名：gt00

バッチ処理システムの使い方

- 主要コマンド（Reedbushの場合）
  - ジョブの投入：
    qsub <ジョブスクリプトファイル名>
  - 自分が投入したジョブの状況確認： rbstat
  - 投入ジョブの削除： qdel <ジョブID>
  - バッチキューの状態を見る： rbstat --rsc
  - バッチキューの詳細構成を見る： rbstat –rsc -x
  - 投げられているジョブ数を見る： rbstat -b
  - 過去の投入履歴を見る： rbstat –H
  - 同時に投入できる数／実行できる数を見る：rbstat --limit

rbstat --rsc の実行画面例

```
$ rbstat --rsc
QUEUE                    STATUS             NODE
u-debug                  [ENABLE ,START]     54
u-short                  [ENABLE ,START]     16
u-regular                [ENABLE ,START]
  |---- u-small          [ENABLE ,START]    288
  |---- u-medium         [ENABLE ,START]    288
  |---- u-large          [ENABLE ,START]    288
  |---- u-x-large        [ENABLE ,START]    288
u-interactive            [ENABLE ,START]
  |---- u-interactive_1  [ENABLE ,START]     54
  |---- u-interactive_4  [ENABLE ,START]     54
u-lecture                [ENABLE ,START]     54
u-lecture8               [DISABLE,START]     54
u-tutorial               [ENABLE ,START]     54
```

使える
キュー名
(リソース
グループ)

現在
利用可能か

利用可能ノード数

rbstat --rsc -x の実行画面例

```
$ rbstat --rsc -x
QUEUE                STATUS           MIN_NODE MAX_NODE MAX_ELAPSE REMAIN_ELAPSE MEM(GB)/NODE  PROJECT
u-debug              [ENABLE ,START]         1       24   00:30:00      00:30:00         244GB  pz0105,gcXX
u-short              [ENABLE ,START]         1        8   02:00:00      02:00:00         244GB  pz0105,gcXX
u-regular            [ENABLE ,START]
  |---- u-small       [ENABLE ,START]         4       16   12:00:00      12:00:00         244GB  gcXX,pz0105
  |---- u-medium      [ENABLE ,START]        17       32   12:00:00      12:00:00         244GB  gcXX
  |---- u-large       [ENABLE ,START]        33       64   12:00:00      12:00:00         244GB  gcXX
  |---- u-x-large     [ENABLE ,START]        65      128   06:00:00      06:00:00         244GB  gcXX
u-interactive        [ENABLE ,START]
  |---- u-interactive_1 [ENABLE ,START]      1        1   00:15:00      00:15:00         244GB  pz0105,gcXX
  |---- u-interactive_4 [ENABLE ,START]      2        4   00:05:00      00:05:00         244GB  pz0105,gcXX
u-lecture            [ENABLE ,START]         1        8   00:10:00      00:10:00         244GB  gt00,gtYY
u-lecture8           [DISABLE,START]         1        8   00:10:00      00:10:00         244GB  gtYY
u-tutorial           [ENABLE ,START]         1        8   00:10:00      00:10:00         244GB  gt00
```

使える
キュー名
(リソース
グループ)

現在
利用可能か

ノードの
実行情報

課金情報(財布)
実習では1つのみ

rbstat --rsc -b の実行画面例

```
$ rbstat --rsc -b
QUEUE                  STATUS           TOTAL RUNNING QUEUED HOLD BEGUN WAIT EXIT TRANSIT NODE
u-debug                [ENABLE ,START]      1       1      0    0     0    0    0       0   54
u-short                [ENABLE ,START]      9       3      5    1     0    0    0       0   16
u-regular              [ENABLE ,START]
  |---- u-small        [ENABLE ,START]     38      10      6   22     0    0    0       0  288
  |---- u-medium       [ENABLE ,START]      2       2      0    0     0    0    0       0  288
  |---- u-large        [ENABLE ,START]      4       2      0    2     0    0    0       0  288
  |---- u-x-large      [ENABLE ,START]      1       0      1    0     0    0    0       0  288
u-interactive          [ENABLE ,START]
  |---- u-interactive_1 [ENABLE ,START]     0       0      0    0     0    0    0       0   54
  |---- u-interactive_4 [ENABLE ,START]     0       0      0    0     0    0    0       0   54
u-lecture              [ENABLE ,START]      0       0      0    0     0    0    0       0   54
u-lecture8             [DISABLE,START]      0       0      0    0     0    0    0       0   54
u-tutorial             [ENABLE ,START]      0       0      0    0     0    0    0       0   54
```

使えるキュー名(リソースグループ)

現在使えるか

ジョブの総数

実行しているジョブの数

待たされているジョブの数

ノードの利用可能数

バッチジョブ実行による標準出力、標準エラー出力

- バッチジョブの実行が終了すると、標準出力ファイルと標準エラー出力ファイルが、ジョブ投入時のディレクトリに作成されます。
- 標準出力ファイルにはジョブ実行中の標準出力、標準エラー出力ファイルにはジョブ実行中のエラーメッセージが出力されます。

ジョブ名.oXXXXX --- 標準出力ファイル
ジョブ名.eXXXXX --- 標準エラー出力ファイル
(XXXXX はジョブ投入時に表示されるジョブのジョブID)

おわり

お疲れさまでした