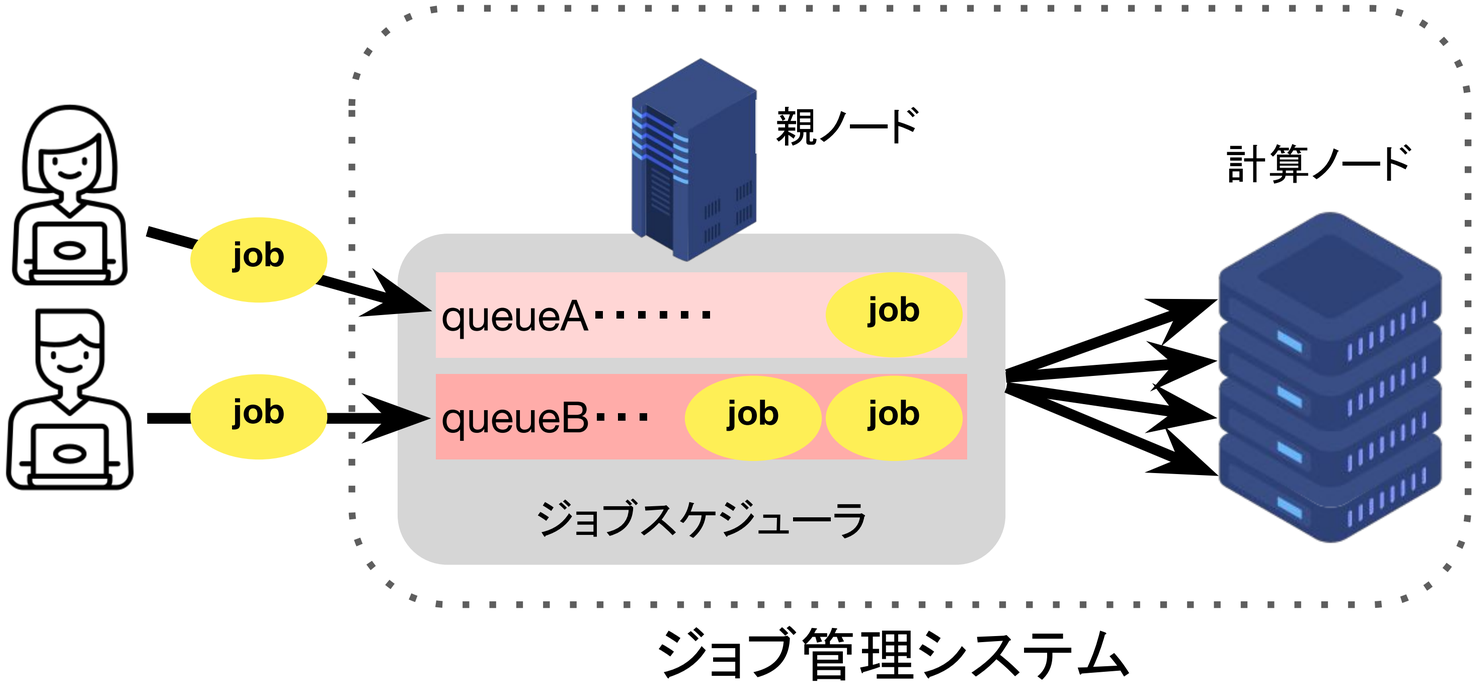
RCCS計算機利用方法

ジョブ管理システム：

・複数の人間が同じ計算機群を使う

・どのマシン/CPUが空いてるか？ どの計算を優先させるべきか？

- 自動で計算機資源の割り当てて効率を上げる
- ユーザはコマンドで親ノードにジョブを投げるだけ
- ジョブは「キュー」と呼ばれるリストに入って実行を待つ



・ジョブ：<u>実行するコマンドやアプリケーションを記述したシェルスクリプト形式のタスク</u>

RCCSにおけるジョブ管理システム

全ての解析はジョブ管理システムを経由して実行すること

・基本はPBSの仕様に従う

・RCCSではジョブ投入、ジョブ確認などに独自コマンドを採用している

コマンド対応表 PBS/RCCS ジョブ管理システム

PBSコマンド	RCCSコマンド	説明
qsub file_name	**jsub** file_name	ジョブを投入する
qstat -u my_account	**jobinfo**	自分のジョブの状態を表示
qstat -Q	**jobinfo -s**	キュー/ジョブタイプの詳細を表示
qstat	**jobinfo -a**	全てのジョブを表示（-g で同グループのみを表示）
qdel *jobID*	**jdel** *jobID*	指定したIDのジョブを削除
qhold *jobID*	**jhold** *jobID*	指定したIDの待機中ジョブを実行されないようにホールド
qrls *jobID*	**jrls** *jobID*	ホールド状態のジョブをリリース（実行待ち状態にする）
tracejob *jobID*	**joblog**	終了したジョブの履歴を表示（CPU点数も表示）

ジョブ実行の流れ

・実行したいコマンドをシェルスクリプトに書く

testjob.sh

```
#!/bin/bash
#PBS -l select=1:ncpus=4:mpiprocs=1:ompthreads=4
#PBS -l walltime=4:00:00
source /apl/bio/etc/bio.sh
module load diamond
cd ${PBS_O_WORKDIR}
diamond makedb --in spo --db spo
diamond blastp --query sce_prot.fasta --db spo.dmnd --out sce-spo.tab
--outfmt 6 --threads ${NCPUS}
```

- **jsub** コマンドにシェルスクリプトのファイルを渡して実行

```
$ jsub testjob.sh
6439875.ccpbs1
```

- **jobinfo** コマンドでジョブの実行状況を確認

```
$ jobinfo
```

シェルスクリプトファイル （ジョブファイル）

```
#!/bin/bash
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12
#PBS -l walltime=4:00:00
source /apl/bio/etc/bio.sh
module load diamond
cd ${PBS_O_WORKDIR}
diamond blastp --db spo.dmnd --out sce.tab --outfmt 6 --query sce_prot.fasta
```

←①シェルの指定

←② jsubオプション

←③module呼び出しとアプリケーションload

←④ジョブ投入ディレクトリへの移動

↑⑤実行するコマンド

1. **シェルの指定**: ジョブスクリプトのシェルの種類を指定
2. **jsubオプション**: 行の先頭を #PBS で始めると、jsubコマンドのオプション指示として処理される
3. **module呼び出しとアプリケーションload**: 前講義を参照のこと
4. **ジョブ投入ディレクトリへの移動** :jsubを実行したディレクトリに移動
   （変数 ${PBS_O_WORKDIR} に jsub実行ディレクトリ名が自動で設定される）
   これをしないとホームディレクトリにいる状態で続くコマンドを実行しようとする
5. **実行するコマンド** ： コマンド、パスの設定、変数のセット等スクリプト本体を記述

ジョブ実行： jsub

- シェルスクリプトを作成

testjob.sh

```
#!/bin/bash
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12
#PBS -l walltime=4:00:00

.....   (略)
```

- jsubコマンドにシェルスクリプトのファイルを渡して実行

```
$ jsub testjob.sh
6439875.ccpbs1
```

- jobinfo コマンドでジョブの実行状況を確認

```
$ jobinfo -c
--------------------------------------------------------------------------------
Queue    Job ID Name              Status CPUs User/Grp     Elaps     Node/(Reason)
--------------------------------------------------------------------------------
H(c)     6439875 testjob.sh        Run     12   ebv/zo0      0:00:20   ccc147
--------------------------------------------------------------------------------
```

ジョブの確認:jobinfo

・-c オプション:最新の状態を表示

```
$ jobinfo -c
-------------------------------------------------------------------------
Queue  Job ID   Name        Status  CPUs User/Grp   Elaps    Node/(Reason)
-------------------------------------------------------------------------
H      6439875  job1.sh       Run    12  ebv/---    0:00:20   ccc147
-------------------------------------------------------------------------
H      6439876  job2.sh       Run    12  ebv/---    0:00:20   ccc147
-------------------------------------------------------------------------
```

・オプションなし:数分古い情報だがjobtype, グループ名も表示

```
$ jobinfo
-------------------------------------------------------------------------
Queue  Job ID   Name        Status  CPUs User/Grp   Elaps    Node/(Reason)
-------------------------------------------------------------------------
H(c)   6439875  job1.sh       Run    12  ebv/zo0    0:00:20   ccc147
-------------------------------------------------------------------------
H(c)   6439876  job2.sh       Run    12  ebv/zo0    0:00:20   ccc147
-------------------------------------------------------------------------
```

自身とグループ使用状況の確認 jobinfo −s

```
$ jobinfo -s
User/Group Stat:
-------------------------------------------------------------------------
 queue: H                              | user(ebv)                | group(zo0)
-------------------------------------------------------------------------
   NJob (Run/Queue/Hold/RunLim) |    1/    0/    0/-    |    1/    0/    0/5000
   CPUs (Run/Queue/Hold/RunLim) |   12/    0/    0/-    |   12/    0/    0/4096
   GPUs (Run/Queue/Hold/RunLim) |    0/    0/    0/-    |    0/    0/    0/20
   core (Run/Queue/Hold/RunLim) |    0/    0/    0/1600 |    0/    0/    0/1600
   lmem (Run/Queue/Hold/RunLim) |   12/    0/    0/-    |    0/    0/    0/896
 sncore (Run/Queue/Hold/RunLim) |    0/    0/    0/-    |    0/    0/    0/4096
-------------------------------------------------------------------------
 queue: HB                             | user(ebv)                | group(zo0)
-------------------------------------------------------------------------
   NJob (Run/Queue/Hold/RunLim) |    0/    0/    0/-    |    0/    0/    0/-
   CPUs (Run/Queue/Hold/RunLim) |    0/    0/    0/-    |    0/    0/    0/-
   GPUs (Run/Queue/Hold/RunLim) |    0/    0/    0/-    |    0/    0/    0/-
-------------------------------------------------------------------------
```

jobinfoオプション（抜粋）

オプション	説明
`--help`	ヘルプを表示
`-c`	最新の情報を表示。−m, −w, −s と同時指定不可
`-s`	キューの空き状況、自身の利用状況を表示。他のオプションと同時指定不可
`-w`	ジョブ投入時の作業ディレクトリを表示　−c と同時指定不可
`-m`	メモリ使用量の表示　−c と同時指定不可
`-L`	実行中ホストの全リストを表示
`-n`	各計算ノードの状況を表示
`-g`	所属グループのユーザーのジョブも表示
`-a`	全ユーザーの情報を表示、ユーザー名やジョブ名などは隠蔽される

ジョブを削除する:jdel

```
$ jdel jobID
```

・ジョブIDの番号を指定して削除（IDはjobifnoなどで確認）

・自分がjsubしたジョブのみ削除可能

・CPU点数はそれまでかかった分消費される

```
$ jdel 6439875
```

ジョブ終了後：.e ファイルと .o ファイル

- ジョブ終了後に自動で作られるファイル

  標準エラー出力の内容　`job_name.ejobID`

  標準出力の内容　　`job_name.ojobID`

- ジョブが実行されたノード上で保持され、ジョブの終了後にjsubされたマシンに送られる
- エラーがあった際の詳細が書かれている

リソース消費量, 残量の確認　showlim/joblog

・これまでにグループで消費したCPU点数を表示

```
$ showlim -c
```

・これまでにグループで消費したCPU点数をメンバーごとに表示

```
$ showlim -c -m
```

・過去のジョブが消費してきた点数を表示

```
$ joblog
```

・グループメンバーのストレージ使用量と残量を表示

```
$ showlim -d -m
```

カウントは年度変わり目でリセットされる