

大規模なデータ解析のためのUNIX入門

# SunGridEngine 使用方法

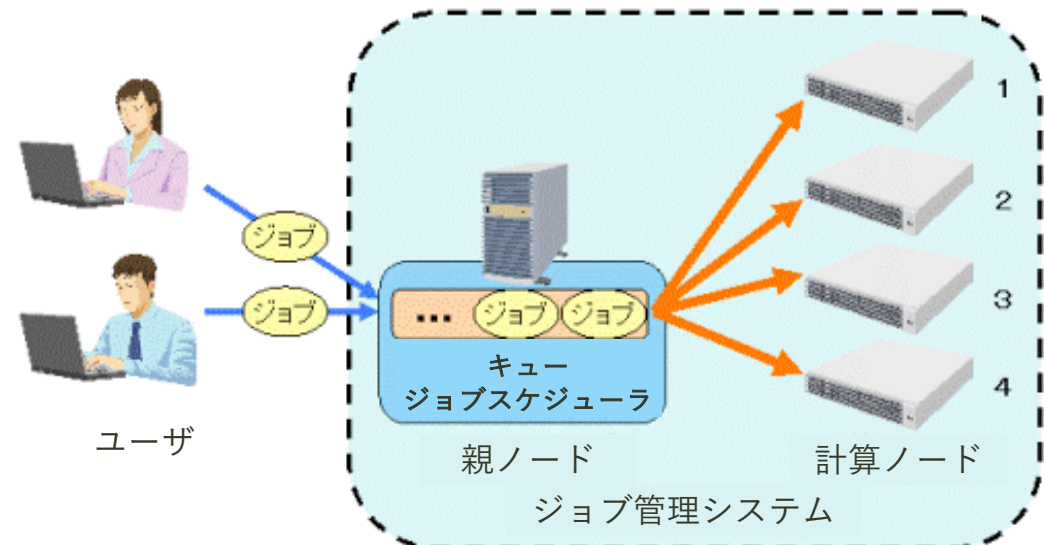
23/June/2017

情報管理解析室 西出 浩世 🐈

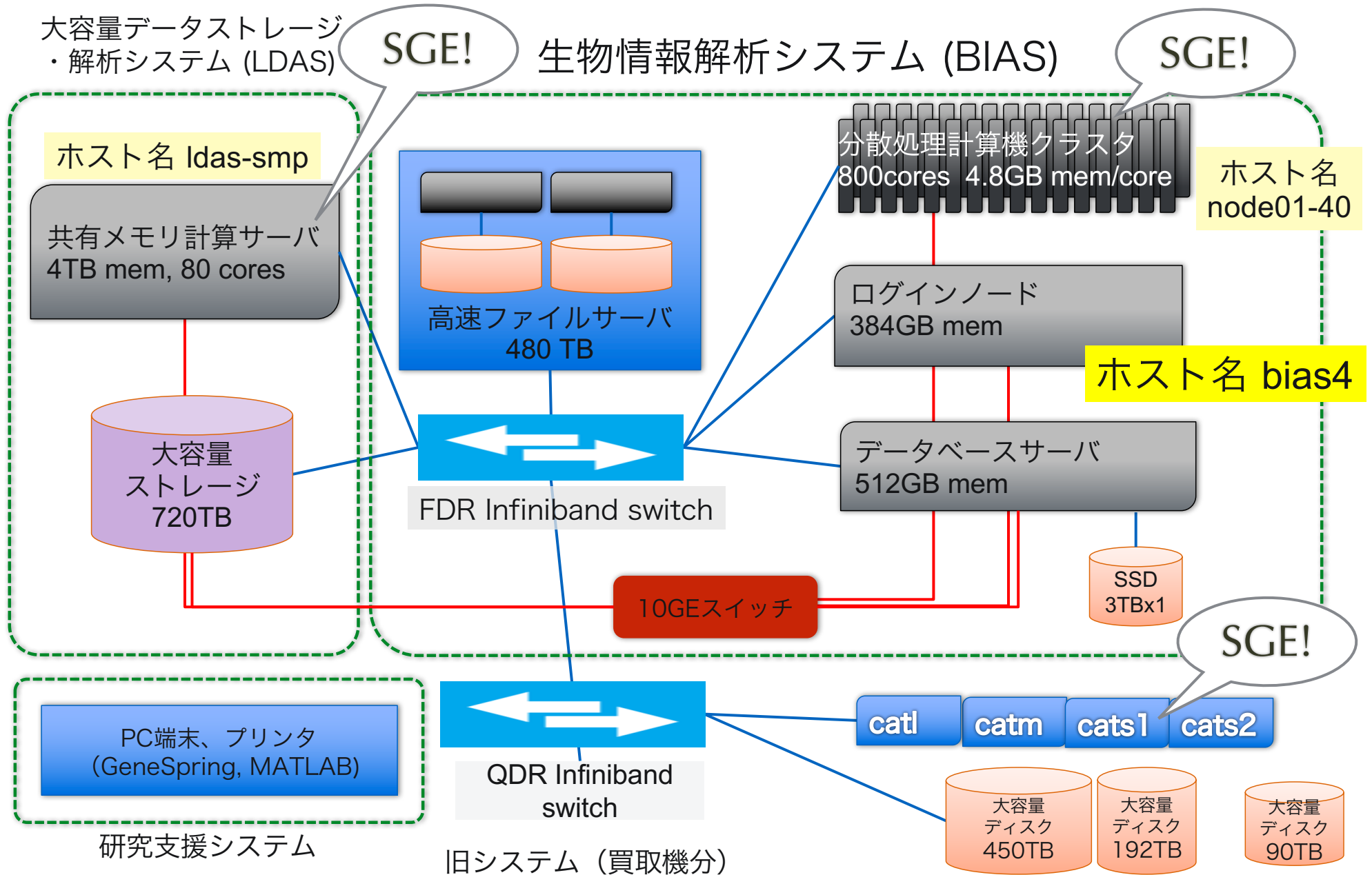
# ジョブ管理システム：SunGridEngine

複数の人間が同じ計算機群を使いたい・・・  
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？

- ▶ 親ノードを設置し、ジョブ管理システムを運用
- ▶ 計算機資源の割り当てを自動で行い、効率を上げる
- ▶ ユーザは親ノードにジョブを投げるだけ（親ノードの名前すら知らなくてもよい）
- ▶ SunGridEngine (SGE)



# 生物情報解析システムの構成



# 作業ディレクトリ

~/data/4\_sge

```
$ cd ~/data/4_sge  
$ ls  
bowtie1.sh  ex.sh  genome  test_fastq
```

genome ディレクトリ、test\_fastq ディレクトリ内の内容を確認

```
$ ls genome  
$ ls test_fastq
```

\* 解析結果を格納するディレクトリ「results」を作成する

```
$ mkdir results
```

# ジョブ管理システムの利用

- 実行したいコマンドをシェルスクリプト内に記述し、qsubコマンド（後述）を用いて実行させる
- シェルスクリプト ex.sh の中身を確認

```
$ less ex.sh
```

ex.sh

```
#!/bin/sh
path=~ /data/4_sge
bowtie2 -x ${path}/genome/ecoli_genome
        -U ${path}/test_fastq/ecoli.1.fastq
        -S ${path}/results/ecoli.1.sam
```

- ~/data/4\_sge/rnaseq 内の ecoli.1.fastq ファイルを、  
~/data/4\_sge/genome にある ecoli\_genome にbowtie2 でマッピングし、結果は ~/data/4\_sge/results に保存

# SGE 実行コマンド qsub

- シェルスクリプトをジョブ管理システム(SGE)に投入：qsub
- **qsub** コマンドの引数にシェルスクリプト名を付けて実行

```
$ qsub scriptname
```

- ex.sh をqsubコマンドで実行

```
$ qsub ex.sh
```

```
Your job 814953 ("ex.sh") has been submitted
```

ジョブ番号

- 投入されたジョブは番号が付けられ、ログインノードから分散処理  
計算機クラスタ内のノードに送られて実行される
- 標準出力と標準エラー出力のログがホームディレクトリにファイル  
として作られる
  - シェルスクリプト名.oジョブ番号 (標準出力)
  - シェルスクリプト名.eジョブ番号 (標準エラー出力)

# qsub (ex.sh) 結果の確認

```
$ cd results
```

```
$ ls
```

```
ecoli.1.sam
```

結果ファイルの確認

```
$ ls ~/ex.sh.*
```

```
ex.sh.e814953 ex.sh.o814953
```

標準出力・標準エラー出力ファイル

```
$ less ~/ex.sh.e814953
```

標準エラー出力ファイルの中身を確認

```
330118 reads; of these:
```

```
  330118 (100.00%) were unpaired; of these:
```

```
    3364 (1.02%) aligned 0 times
```

```
   229054 (69.39%) aligned exactly 1 time
```

```
    97700 (29.60%) aligned >1 times
```

```
98.98% overall alignment rate
```

```
$ less ecoli.sam
```

結果ファイルの中身を確認

```
$ cd ..
```

作業ディレクトリに戻る

# SGE その他コマンド

<b>qsub</b> <i>script_filename</i>	ジョブを投入
<b>qstat</b>	自分のジョブの状態を表示
<b>qstat -u '*'</b>	全ユーザのジョブ状態を表示
<b>qdel</b> <i>job-ID</i>	ジョブを削除

```
$ qstat
```

```
job-ID  prior  name      user  state submit/start at  queue slots ja-task-ID
-----
814953  0.00000 bowtie.sh  hiroyo  r      01/08/2015 14:14:54      1
814954  0.00000 bowtie.sh  hiroyo  qw     01/08/2015 14:14:54      1
814955  0.00000 bowtie.sh  hiroyo  qw     01/08/2015 14:14:54      1
```

```
$ qdel 814953
```

```
hiroyo has deleted job 814953
```



# qsub のオプション

- qsub には様々なオプションがあり、シェルスクリプト内で「#\$」に続けて書いておくことで機能を加えることができる

```
$ less bowtie1.sh
```

bowtie1.sh

```
#!/bin/sh
```

```
#$ -q gitc
```

gitcキューを指定

```
#$ -cwd
```

qsubしたディレクトリに移動してジョブを実行

```
bowtie2 -x ../rnaseq/ecoli_genome
```

```
-U ../rnaseq/test_fastq/ecoli.1.fastq
```

```
-S results/ecoli.1.sam
```

- cwd を指定しているので、データ等を相対パスで書いている
- この場合標準出力・エラー出力ログも同じ場所にできる

# qsub の主なオプション

オプション	説明
<b>#\$ -o filename</b>	標準出力の結果を指定したファイルに保存
<b>#\$ -e filename</b>	標準エラー出力の結果を指定したファイルに保存
<b>#\$ -q queue_name</b>	キューを指定してジョブを実行
<b>#\$ -cwd</b>	qsubした時のディレクトリに移動してジョブを実行
<b>#\$ -v 環境変数</b>	環境変数をジョブに渡す
<b>#\$ -N job_name</b>	ジョブ名を指定する
<b>#\$ -s shell_name</b>	ジョブスクリプトを指定したシェルで実行
<b>#\$ -a MMDDhhmm</b>	ジョブの開始日時を指定
<b>#\$ -l resource_name 値</b>	ジョブが使うリソース量を指定する
<b>#\$ -pe PE_name プロセス数</b>	並列ジョブを実行する場合の環境と並列数の指定
<b>#\$ -t 開始番号-終了番号</b>	アレイジョブを実行

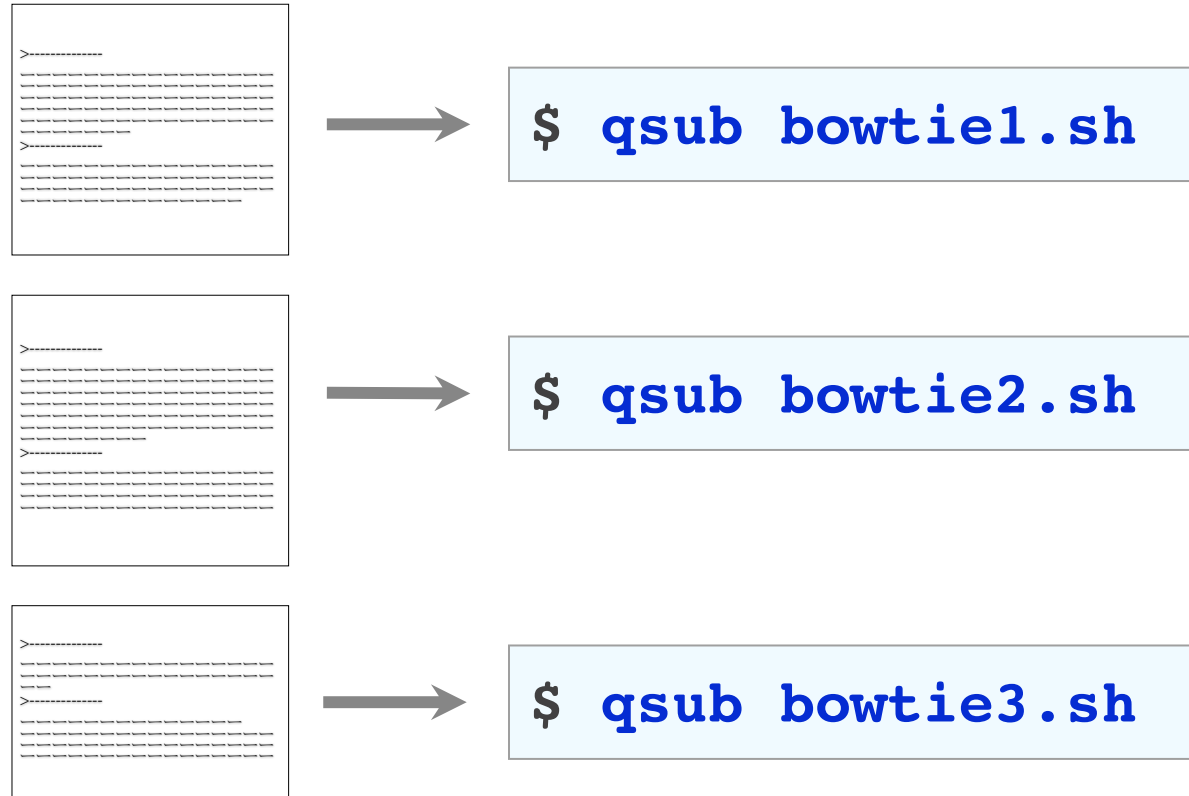
当システムにおける

## キュー（ジョブの待ち行列）構成

	分散並列処理型			共有メモリ型		
	分散処理計算機クラスタ			共有メモリ型計算サーバ		
キュー名	small	medium	large	smps	smpm	smpl
ジョブの特徴	短時間・並列多	中規模	長時間	中メモリ	大メモリ	最大メモリ
利用ノード	node01-40	node01-40	node01-40	ldas-smp	ldas-smp	ldas-smp
最大実行時間/job	6時間	72時間	no limit	no limit	no limit	no limit
最大ジョブ数/キュー	580	200	20	8	4	1
最大使用メモリ/ジョブ	4GB	4GB	4GB	500GB	1TB	4TB
利用できるPE	smp, mpi128, mpi256, make	smp, mpi128, mpi256, make	smp, mpi128, make	smp	smp	smp

- キューを指定しない場合、デフォルトでは「small」で実行されます
- ユーザあたりジョブ同時実行数は最大 400 です
- 今回の講習では特別キュー「gitc」を使っています。講義中しか使えません

# 並列化



- 1台では時間がかかる計算も、分割して複数台、複数CPUに仕事をさせれば数倍の速度で終わる
- 一つのスクリプトに複数の処理を併記しても「同じ計算機上で順次実行されるだけ」

# 並列実行

- emacs で bowtie1.sh を書き換えて、「bowtie2.sh」として保存する  
emacs : 別名で保存 (C-x, C-w)

```
$ emacs bowtie1.sh
```

```
#!/bin/sh  
#$ -q gitc  
#$ -cwd  
bowtie2 -x genome/ecoli_genome  
        -U test_fastq/ecoli.2.fastq  
        -S results/ecoli.2.sam
```

bowtie2.sh

- bowtie2.sh = ecoli.2.fastq をマッピング
- 書き換えたらqsubでSGEに両方を投入し、qstat で実行状況を確認

```
$ qsub bowtie1.sh  
$ qsub bowtie2.sh  
$ qstat
```

# 結果ファイルの確認

```
$ cd results
```

```
$ ls
```

```
ecoli.sam ecoli.1.sam ecoli.2.sam
```

```
$ less ecoli.2.sam
```

```
$ cd ..
```

# アレイジョブ

- ✓ test\_fastq には ecoli.1.fastq ~ ecoli.12.fastq : 12個のfastqファイルがある
- ✓ 似たようなジョブファイルをいくつも作るのは面倒
  - SGE のオプションである アレイジョブ機能 を使う
  - 1つのスクリプトファイルで複数の独立ジョブを実行させる
  - **#\$ -t** 開始番号-終了番号 でアレイジョブの数を指定
  - 変数 `${SGE_TASK_ID}` が自動生成される
  - `${SGE_TASK_ID}` にアレイジョブ数がセットされ、インクリメント（コンピュータ用語では変数を一つ増やすこと）しながら回数分実行される
  - qsub は1回でよい

# アレイジョブ用のスクリプトを用意

- emacsでbowtie1.shを改造し、bowtie\_all.sh として保存

```
$ emacs bowtie1.sh
```

```
#!/bin/sh  
#$ -q gitc  
#$ -cwd  
#$ -t 1-12
```

```
bowtie2 -x genome/ecoli_genome  
        -U test_fastq/ecoli.${SGE_TASK_ID}.fastq  
        -S results/ecoli.${SGE_TASK_ID}.sam
```

bowtie\_all.sh



# アレイジョブ実行

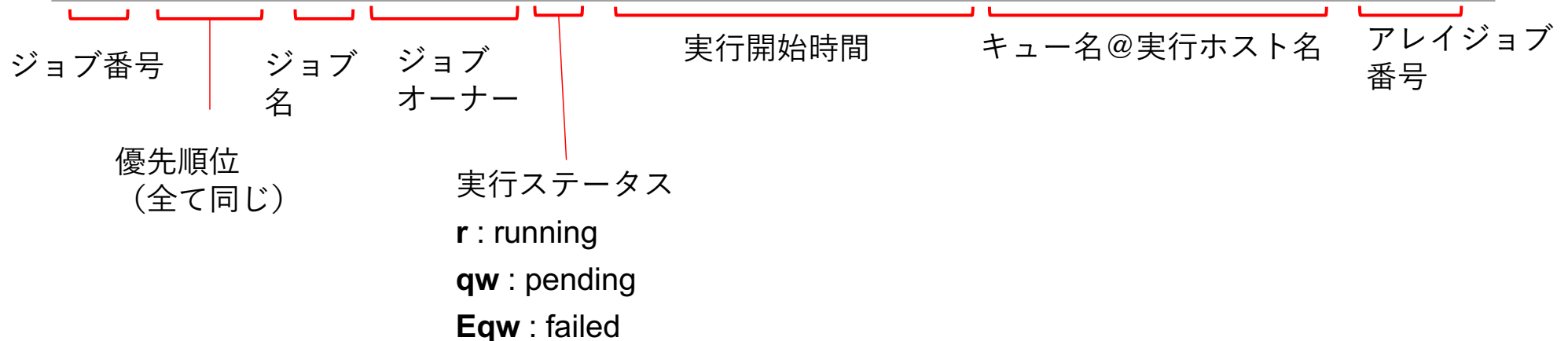
- 実行とジョブの確認

```
$ qsub bowtie_all.sh
```

```
Your job 814953 ("bowtie_all.sh") has been submitted
```

```
$ qstat
```

job-ID	prior	name	user	state	submit/start at	queue	slots	ja-task-ID
6597	0.5050	1job	hiroyo	r	01/19/2017 18:04:04	<a href="#">gitc@node03.local</a>	1	1
6597	0.5050	1job	hiroyo	r	01/19/2017 18:04:04	<a href="#">gitc@node06.local</a>	1	2
6597	0.5050	1job	hiroyo	r	01/19/2017 18:04:04	<a href="#">gitc@node08.local</a>	1	3
6597	0.5050	1job	hiroyo	r	01/19/2017 18:04:04	<a href="#">gitc@node06.local</a>	1	4
6597	0.5050	1job	hiroyo	r	01/19/2017 18:04:04	<a href="#">gitc@node08.local</a>	1	5



# 結果ファイルの確認

```
$ cd results
```

```
$ ls
```

```
ecoli.1.sam  ecoli.2.sam  ecoli.3.sam  ecoli.4.sam  
ecoli.5.sam  ecoli.6.sam  ecoli.7.sam  ecoli.8.sam  
ecoli.9.sam  ecoli.10.sam ecoli.11.sam ecoli.12.sam
```

```
$ less ecoli.12.sam
```

```
$ cd ..
```

```
$ ls bowtie_all.*
```

```
bowtie_all.e814953.1 bowtie_all.sh.e814953.2  
bowtie_all.sh.e814953.3 bowtie_all.sh.o814953.4  
bowtie_all.sh.o814953.4 bowtie_all.sh.o814953.1  
bowtie_all.sh.o814953.2 bowtie_all.sh.o814953.3  
bowtie_all.sh.o814953.4 bowtie_all.o814953.5 . . . . .
```

- アレイジョブでは、スクリプト名.ジョブ番号.アレイ番号 のログが作られる

# 演習

- アレイジョブ機能を使って、~/data/4\_sge/results 下の12個のsamファイルについて、htseq-count を使って ecoli.gtf にある遺伝子ごとのヒット数をカウントする。結果は、~/data/4\_sge/results/ecoli.1.htseq ~ ecoli.12.htseq に保存しよう。
  - スクリプトファイル名は htseq.sh とする
  - 遺伝子情報が載った gtf ファイルは  
~/data/3\_ngs/ecoli.gtf にある

# 回答

htseq.sh

```
#!/bin/sh
#$ -q gitc
#$ -cwd
#$ -t 1-12

htseq-count results/ecoli.${SGE_TASK_ID}.sam
    ../3_ngs/ecoli.gtf >
    results/ecoli.${SGE_TASK_ID}.htseq
```

```
$ qsub htseq.sh
```

```
Your job 814953 ("htseq.sh") has been submitted
```