

2013/11/26

# 2013年度 ゲノム支援情報解析講習会

- DDBJへの接続方法、バッチジョブ-

# スパコン使用方法(イメージ)

- ①ゲートウェイノード(gw.ddbj.nig.ac.jp)にログインする
- ②qloginを実行しインタラクティブノードにログインする
- ③qloginしたホストからバッチジョブ投入する
- ④バッチジョブは負荷の低いノードでジョブを実行する
- ⑤ジョブ実行結果をホームディレクトリに出力する
- ⑥ジョブ実行結果を確認する



# バッチジョブとは

計算処理のファイルを集め、一括して処理することをバッチ処理といい、集められたデータ、あるいはそのデータを処理するための一連の手続のことをバッチジョブといいます。

コンピュータはバッチファイルを参照し一連の処理を自動で行います。

# バッチジョブとは



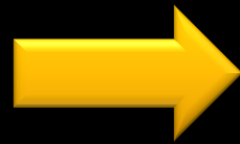
2スロット使用



3スロット使用



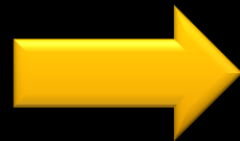
5スロット使用



投入された順番、優先度に応じて空いている計算リソースを見つけ処理を実行する。

# バッチジョブとは

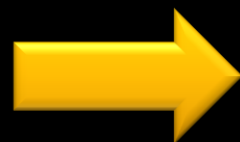
- ① 2スロット使用 (終了)
- ② 3スロット使用
- ③ 5スロット使用
- ④ 6スロット使用



投入された順番、優先度に応じて空いている計算リソースを見つけ処理を実行する。

# バッチジョブとは

- ① 2スロット使用 (終了)
- ② 3スロット使用
- ③ 5スロット使用
- ④ 6スロット使用
- ⑤ 4スロット使用 (待機)



計算リソースが足りない場合は待機状態になり、リソースが空き次第実行される。

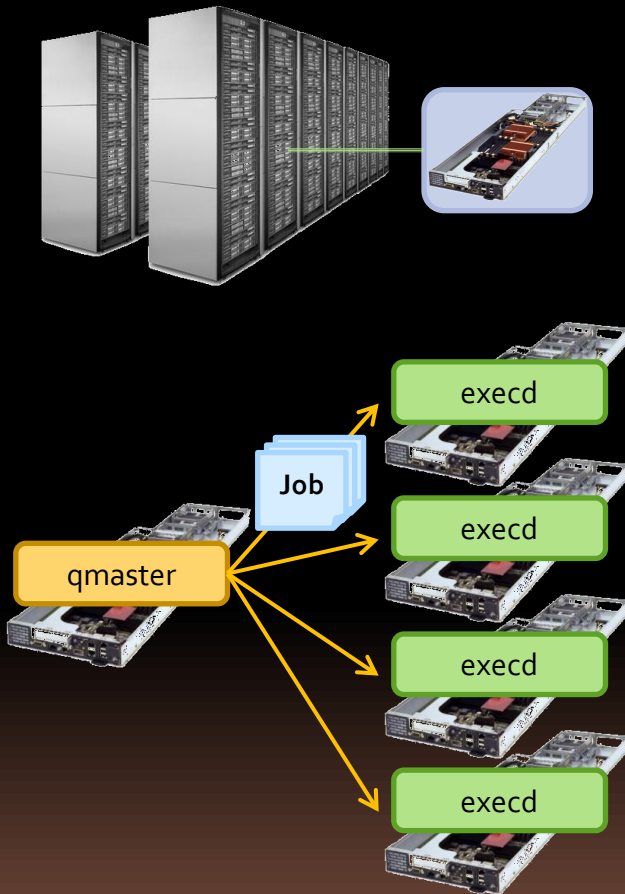
## バッチジョブを利用する利点

- 大量のジョブを逐次、円滑に実行できる
- 複数のユーザが同時に大量のジョブを投入しても、UGEがスケジューリングを行う
- ジョブが求めるメモリ、CPU等に応じて、適切なスケジューリングを行う

## バッチジョブを利用するうえでの注意点

- ジョブの不適切な並列化は行わない
- ジョブ投入時のリソース要求宣言を適切に行わない場合、大規模な計算機のハングアップを招く場合がある

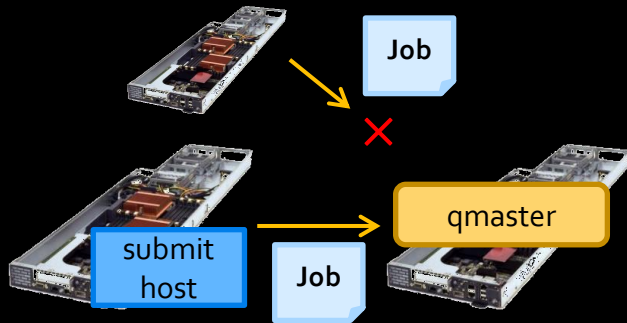
# 基本用語(概念) 1



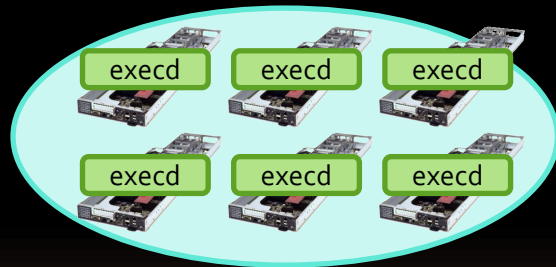
- ホスト（ノード）  
物理的に存在する計算機
- マスターホスト  
UGEのマスターデーモン(qmaster)が動作するホスト  
マスターデーモンはUGEを統括するデーモンで、ジョブの受付、スケジューリング、実行ホストへの配送、回収などを行う
- 実行ホスト  
UGEの実行デーモン(execd)が動作するホスト  
実行デーモンはマスターデーモンからのジョブ実行の指示を受け、ジョブを実行する



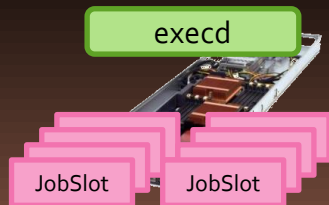
# 基本用語(概念)2



- サブミットホスト  
UGEにジョブを投入可能なホスト。  
qloginコマンドでログイン可能な実行ホストがこれに該当する



- キュー  
ジョブの投入対象。複数の実行ホストで構成される。用途に応じて数種類のキューが存在する



- ジョブスロット  
各実行ホストに設定された、ジョブを実行するための「入れ物」。ジョブはキューに投入され、最終的にスロットに収まる

# キューの種類(11/26時点)

キュー名	ジョブ スロット数	実行時間 の上限	用途など
week_hdd.q	1600	14日	キュー・リソースを指定しない場合、ジョブはこのキューに投入される
week_ssd.q	832	14日	ssdを使用する、短い時間で終了する見込みのジョブを実行する場合に使用
month_hdd.q	96	62日	実行時間が長くなる見込みのジョブを実行する場合に使用
month_ssd.q	64	62日	ssdを使用する、実行時間が長くなる見込みのジョブを実行する場合に使用
month_gpu.q	992	62日	gpuを使用するジョブを実行する場合に使用
month_medium.q	160	62日	mediumノードを使用するジョブを実行する場合に使用
month_fat.q	768	62日	fatノードを使用するジョブを実行する場合に使用
debug.q	64	1日	ジョブの動作確認をする場合使用
login.q	192	—	ジョブの投入を行うために使用

# 実行時間の上限

- 混雑時の実行待ちジョブの渋滞解消を目的として、実行時間の上限を設定している。実行時間の上限を超えたジョブはkillされる
- 「実行時間」は、ジョブが実行されてからの実時間でカウントされる（CPU使用時間等ではない。キューで待機している時間はカウントされない）
- ジョブを投入する前に、動作確認用の環境を使用して実行時間を把握する必要がある

# qlogin

ジョブを投入する場合は、ゲートウェイホストからqloginコマンドでlogin.qのリソースに余裕のあるホストにログインする

```
$ qlogin
Your job 329 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 329 has been successfully scheduled.
Establishing builtin session to host t217i ...
$ uname -n
t217
```

実行ホストにログインするときは必ずqloginコマンドを使用する  
負荷分散機構が適切に機能しなくなるため、直接ログインしてジョブを実行しないこと  
(※直接ログインしているユーザは記録されている)

# ジョブの投入1

ジョブは、UGE向けに記述したシェルスクリプトを作成して投入する

- サンプルファイル -

```
$ cp /tmp/kawagoe/test.sh /home/[USER]/
```

```
#!/bin/sh
#$ -S /bin/sh

pwd
hostname
date
sleep 20
date
echo "to stderr" 1>&2
```

2行目の“#\$”は、UGEオプションを指定するための接頭辞  
“#\$ -S”で、このシェルスクリプトがUGE上で動作する際に使用するインタプリタを指定する(この例の場合、インタプリタは/bin/sh)  
この行を省略した場合、ジョブ投入時のコマンドオプションで  
“-S 使用するインタプリタのパス”を指定する必要がある

# ジョブの投入2

qsubコマンドでジョブを投入する

```
$ qsub test.sh
```

ジョブを投入すると、実行待ち行列にジョブが入る

投入したジョブの状況は、qstatコマンド(後述)で確認する

実行後、ジョブの出力を確認する

ホームディレクトリに、ジョブの標準出力、標準エラー出力を記録したファイルが出力される

```
$ cat ~/test.sh.o325
/lustrel1/home/ddbjuser
t165
2013年 11月 20日 水曜日 11:15:01 JST
2013年 11月 20日 水曜日 11:15:21 JST
$ cat ~/test.sh.e325
to stderr
```

# qsubの主なオプション1

-s <インタプリタのパス>

スクリプトファイルを実行する際のインタプリタのパスを指定する  
シェル以外に、Perl,Ruby等のスクリプト言語のインタプリタも指定できる  
例： (shを指定): -S /bin/sh (Perlを指定): -S /usr/local/bin/perl

-cwd

ホームディレクトリではなく、qsubコマンド実行時のディレクトリでジョブ  
を実行する。このオプションを指定した場合、標準出力および標準エラー出力  
ファイルは、qsubコマンド実行時のディレクトリに出力される

-o <標準出力の出力先> -e <標準エラー出力の出力先>

ジョブの標準出力および標準エラー出力の出力先を指定する  
標準出力または標準エラー出力をファイル出力しない場合は出力先に  
“/dev/null”を指定する  
例： -o /dev/null -e /dev/null

# qsubの主なオプション2

-N <ジョブの別名>

qstat等で確認可能なジョブの名前を、指定した名前に変更する  
指定しない場合、ジョブの名前はスクリプト名と同じとなる

-l リソース要求1, リソース要求2, ...

-l リソース要求1 -l リソース要求2 -l ...

主にキューの選択、メモリ利用上限の変更に使う  
詳細は後述する



# ジョブの状況確認

投入したジョブの状況は`qstat`コマンドで確認する

ジョブが待ち行列に入っている場合、stateに”qw”が表示される

```
$ qstat
job-ID  prior    name          user          state submit/start at
-----
      325 0.00000 test.sh       ddbjuser      qw    03/19/2012 19:11:56
```

ジョブが実行中の場合、stateに”r”が表示される

```
$ qstat
job-ID  prior    name          user          state submit/start at
-----
      325 0.00000 test.sh       ddbjuser      r     03/19/2012 19:11:56
```

主なstateは以下の通り。場合によっては複数組み合わせで表示される

r	ジョブは実行ホスト上で実行中です
qw	ジョブはキューで待機しています
t	ジョブは実行ホストへ転送処理中です
E	ジョブにエラーが発生しています
d	ジョブは削除処理中です

# qstatの主なオプション

-f

キューの利用状況を合わせて表示する

例： `qstat -f`

-u [uid]

指定した[uid]のジョブも表示する。「\*」とすると、全ユーザのジョブを表示する

例： `qstat -u *`

-j [jobid]

指定した[jobid]のジョブの詳細情報を確認する。エラーステータス“Eqw”となった理由を確認できる。

例： `qstat -j 325`

# ジョブの削除

ジョブを削除する場合、`qdel`コマンドを使用する  
ジョブの削除は、ジョブIDまたはUIDを指定して行う

ジョブIDを指定する場合(ジョブIDのみを指定する)

```
$ qsub test.sh
Your job 326 ("test.sh") has been submitted
$ qdel 326
ddbuser has deleted job 326
```

UIDを指定する場合(“-u” オプションを使用し、UIDを指定する)

```
$ qsub test.sh
Your job 327 ("test.sh") has been submitted
$ qsub test.sh
Your job 328 ("test.sh") has been submitted
$ qdel -u ddbuser
ddbuser has registered the job 327 for deletion
ddbuser has registered the job 328 for deletion
```

# ジョブの実行結果確認

実行が終了したジョブの詳細は`qacct`コマンドで確認する  
ジョブが実際に消費したリソース等が確認できる

```
$ qacct -j 325
```

```
=====
qname          week_hdd.q
hostname       t165i
group          se
owner          ddbjuser
project        NONE
(※中略※)
cpu            0.032
mem            0.001
io             0.000
iow            0.000
maxvmem        208.207M
arid           undefined
```

# ジョブ投入前の注意事項

- ※大量にジョブを投入する前に必ずテストする
  - メモリ枯渇で大量のホストがハングアップする
  - 大量のエラージョブはUGEを過負荷にする
- ※入力ファイル・最終出力のファイルを/tmp, /ssdのような各ホストローカルのディレクトリに配置・出力しない
  - ジョブが実行されるホストで入力を読み込めない
  - 実行後に結果を参照できない
- ※1ジョブで同時実行するプロセスは1プロセスとする。  
(def\_slot(後述)を使わずに、1ジョブで複数プロセスをフォークしない)
- ※1プロセスのスレッド数は1スレッドとする  
(def\_slot(後述)を使わずに、プロセスをマルチスレッドで実行しない)
  - 負荷分散が適切に実施できず、ホストがハングアップする

# キューの使い分け方法1

キューは、” -l ” オプションによるリソース指定により使い分けができる

リソースを指定しない

```
$ qsub test.sh
```

week\_hdd.q, week\_ssd.q が使われる  
優先順位は week\_hdd.q > week\_ssd.q

“month” を指定する (※長い計算時間が見込まれる場合に指定)

```
$ qsub -l month test.sh
```

month\_hdd.q, month\_ssd.q, month\_gpu.q が使われる  
優先順位は month\_hdd.q > month\_ssd.q > month\_gpu.q

“ssd” を指定する (※SSDを使うジョブを投入する場合に指定)

```
$ qsub -l ssd test.sh
```

week\_ssd.q のみが使われる

# キューの使い分け方法2

“month” と “ssd” を指定する  
(※SSDを使うジョブで、長い計算時間が見込まれる場合に指定)

```
$ qsub -l month -l ssd test.sh
```

month\_ssd.q, month\_gpu.qが使われる  
優先順位は month\_ssd.q > month\_gpu.q

“month” と “gpu” を指定する  
(※GPUを使うジョブを投入する場合に指定)

```
$ qsub -l month -l gpu test.sh
```

month\_gpu.qのみが使われる  
※GPU搭載ホストを使う場合は必ず “-l month” を指定する  
※GPUを要求するジョブは1台のGPU搭載ホストで同時に1ジョブのみ動作可能

“month” と “medium” を指定する  
(※Mediumノードを使うジョブを投入する場合に指定)

```
$ qsub -l month -l medium test.sh
```

month\_medium.qのみが使われる  
※Mediumノードを使う場合は必ず “-l month” を指定する

# キューの使い分け方法3

“month” と “fat” を指定する  
(※ Fatノードを使うジョブを投入する場合に指定)

```
$ qsub -l month -l fat test.sh
```

month\_fat.qのみが使われる  
※Fatノードを使う場合は必ず“-l month”を指定する

“debug” を指定する  
(※ジョブの動作確認を行う場合に指定)

```
$ qsub -l debug test.sh
```

debug.qが使われる

“debug” と “gpu” を指定する  
(※GPUを使うジョブの動作確認を行う場合に指定)

```
$ qsub -l debug -l gpu test.sh
```

debug.q内のGPU搭載ホストが使われる



# キューの使い分け方法4

※注意※

GPU、Mediumノード、Fatノードを使いたい場合、それらを使うためのリソース指定（“`gpu`”，“`medium`”，“`fat`”）以外に“`month`”を必ず指定する

現在のキュー構成ではGPUノード、Mediumノード、Fatノードはすべて長時間計算向けのキューにのみ割り当てられているため、それらのキューを使うためには“`month`”のリソース指定が必要となる  
“`month`”の指定がない場合、現在のキューにはリソース指定条件に該当するリソースがないためサブミットは正常に行われるがジョブは実行されない

# 大量のメモリを使用する場合1

- UGEジョブが利用可能なメモリ量は、デフォルトでは4GBに制限されている
- 大容量メモリを使用する場合は利用時に“-l”オプションでメモリ利用量を宣言する

## 1ジョブで8GBのメモリを使用する場合

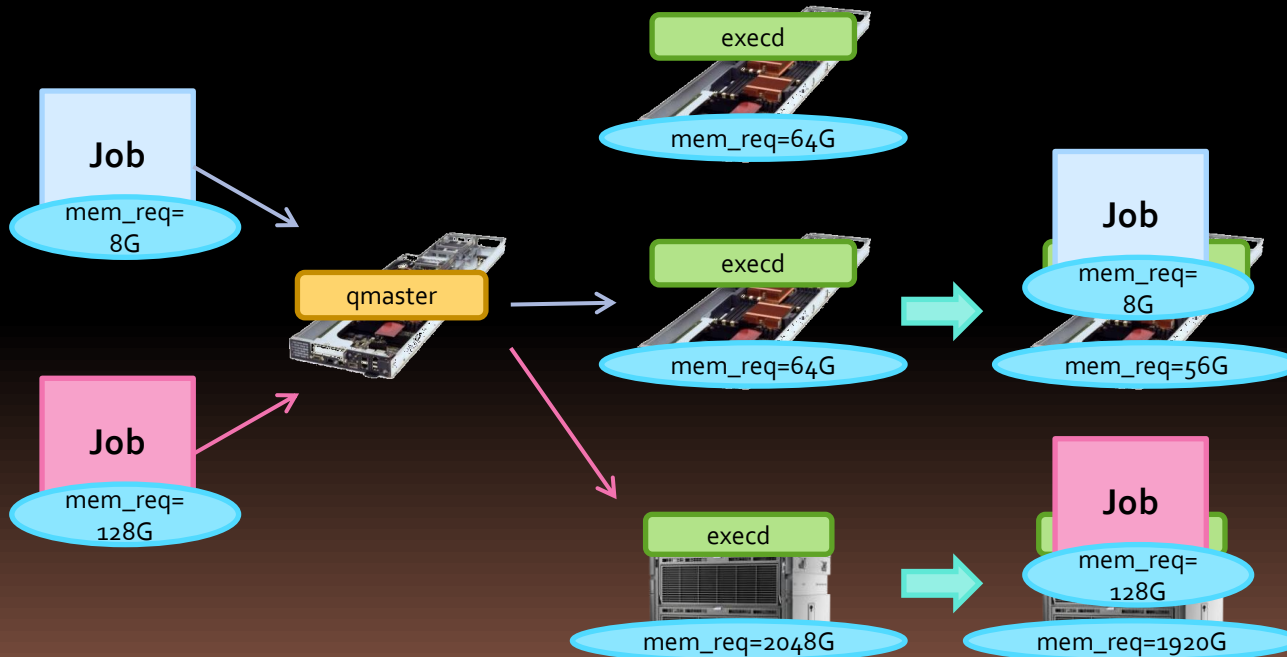
```
$ qsub -l s_vmem=8G -l mem_req=8G test.sh
```

## Mediumノード上で、1ジョブで128GBのメモリを使用する場合

```
$ qsub -l s_vmem=128G -l mem_req=128G -l month -l medium test.sh
```

# 大量のメモリを使用する場合2

s\_vmem: ジョブが使用可能な仮想メモリの上限值を宣言する。ジョブは、ここで指定した量を超えるメモリは使用できない  
mem\_req: 使用するメモリの量を宣言する。実行ホストにはメモリの残容量を表す指標として“mem\_req”の値が設定されており、ジョブの実行状況により増減する。負荷分散の指標の一つとして使われている



1. ジョブ実行中はジョブで宣言された分だけホストの mem\_req値が減る
2. ジョブが終了するとジョブで宣言された分の mem\_req は元に戻る
3. ジョブで宣言された mem\_reqよりホストの mem\_req値が低い場合、そのホストでジョブは実行されない

# 並列環境def\_slotの使用1

複数プロセスをフォークするジョブ、マルチスレッドのプロセスを実行するジョブ等、そのまま投入した場合に過負荷を引き起こす可能性のあるジョブを投入する場合に使用する

```
$ qsub -pe def_slot 2 test.sh
```

“def\_slot” に続く値で、このジョブが消費するジョブスロット数を再定義する

この例の場合、このジョブはジョブスロットを2つ消費する

対象となるジョブ内で同時起動されるプロセスの最大数、ジョブ内で起動されるプロセスが使用する最大スレッド数を指定する値の目安とする

# 並列環境def\_slotの使用2

※注意※

def\_slotを指定した場合、リソース要求の量は  
“-lで指定したリソース量” × “def\_slotで指定したスロット数”  
となる

意図せず過剰なリソース要求を行ってしまう可能性があるので注意。

以下のオプションを指定した場合、リソース要求量は32GBとなる

```
$ qsub -pe def_slot 4 -l max_vmem=8G -l mem_req=8G test.sh
```

リソース要求を明示しない場合、デフォルト値が適用されるので  
以下の場合リソース要求量は16GBとなる

```
$ qsub -pe def_slot 4 test.sh
```

以下の場合リソース要求量は80GBとなるが、Thinノードには  
条件を満たすノードはないため、サブミットされてもジョブは実行されない

```
$ qsub -pe def_slot 10 -l max_vmem=8G -l mem_req=8G test.sh
```

# スパコンへのデータ転送①

- データ転送にはSCPやSFTPを利用する方法とAsperaを利用する方法がある。(FTPは利用できない)

## ①SCP、SFTPの特徴

- ・セキュアなデータ転送が可能である。
- ・WindowsではWinSCP等のフリーソフトを利用する。
- ・Macでは標準搭載機能である。

## ②Asperaの特徴

- ・データ転送がSFTPと比べて高速である。
- ・Aspera Connect(無償)をインストールして使用する。

# スパコンへのデータ転送②

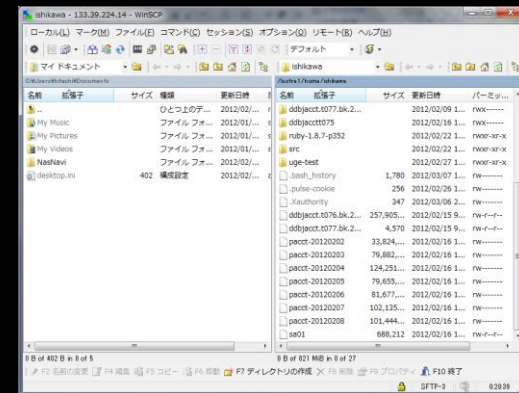


- Windowsからのデータ転送

- ①SFTPを利用する



WinSCPを使用する



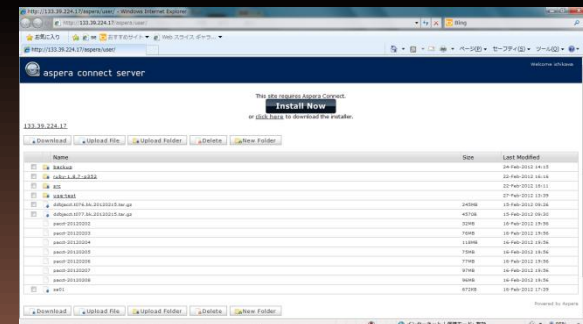
- ②Asperaを利用する



AsperaConnectを使用する

Web UIを使用する

コマンドプロンプトを使用する



# スパコンへのデータ転送③



- Macからのデータ転送

- ①SCPを利用する



ターミナルから転送する

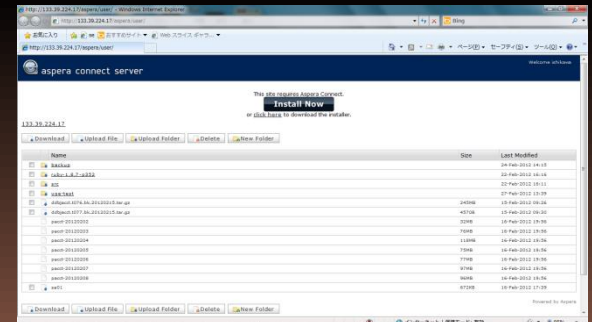


- ②Asperaを利用する



AsperaConnectを使用する

Web UIを使用する  
ターミナルを使用する



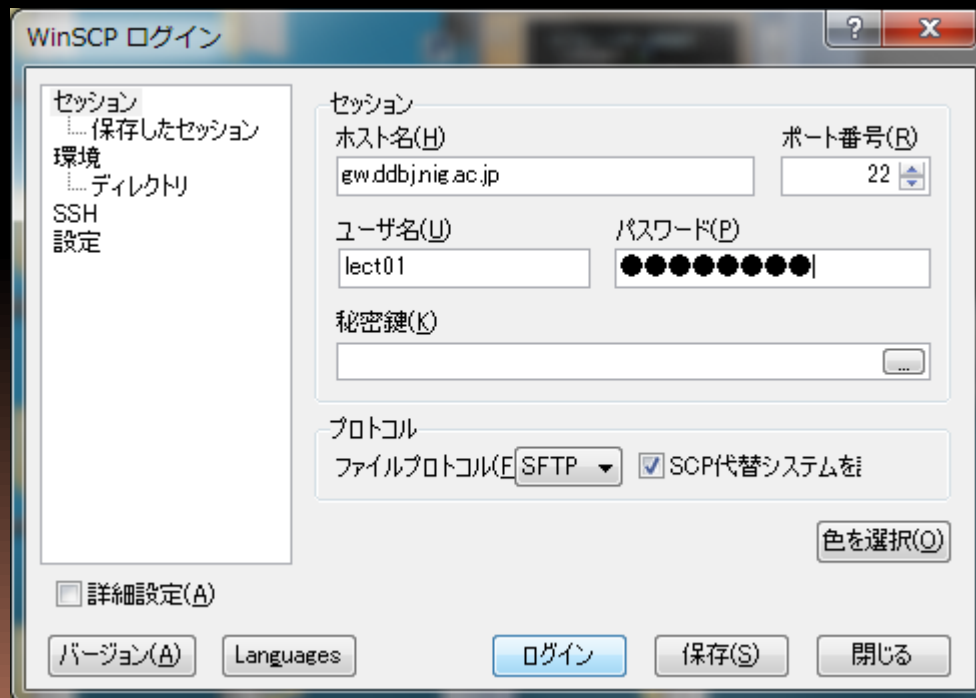


# スパコンへのデータ転送④



## WinSCPを使用する

- ① WinSCPを実行し、セッションを選択する。ホスト名欄に `gw.ddbj.nig.ac.jp` を入力する。ユーザ名およびパスワードを記入する。プロトコルは **SFTP** を選択し、ログインボタンを押下する。

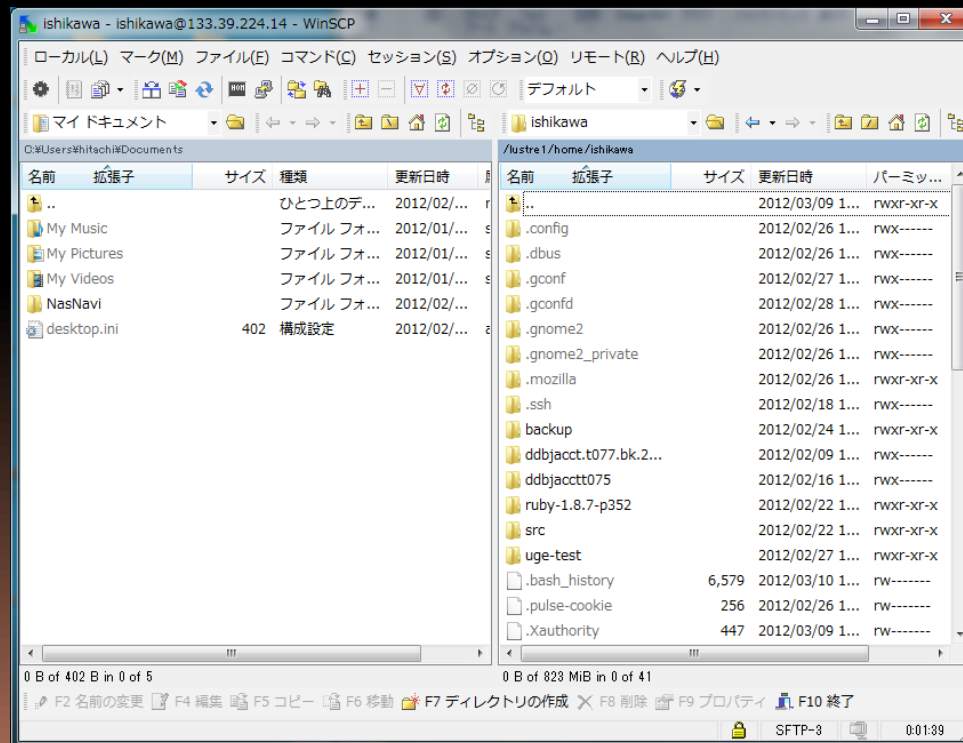


# スパコンへのデータ転送④



## WinSCPを使用する

- ② 左に自分のシステム、右にスパコンのホームディレクトリが表示される。転送したいデータをドラッグ&ドロップすることで転送される。

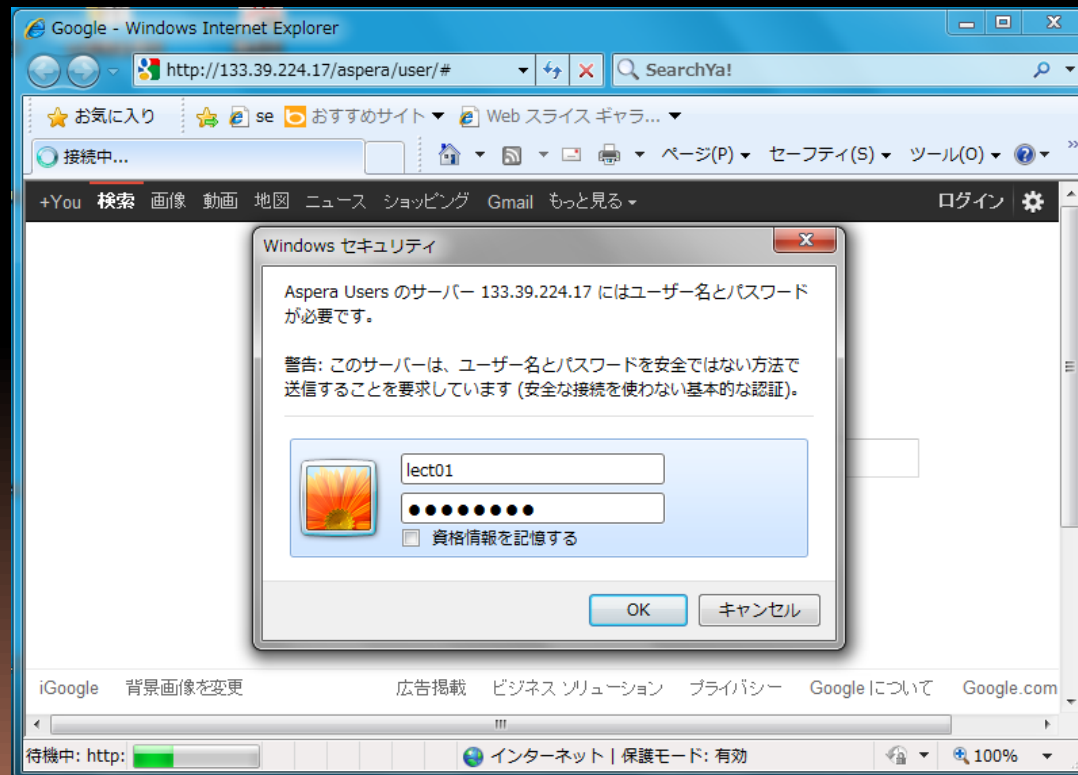


# スパコンへのデータ転送⑤



## Aspera Connectを使用する(Web UI)

- ① ブラウザを開き、<https://ascp.ddbj.nig.ac.jp/aspera/user> にアクセスする。ユーザー名とパスワードを入力し、OKボタンを押下する。



# スパコンへのデータ転送⑤



## Aspera Connectを使用する(Web UI)

②ログイン後、スパコンのホームディレクトリが表示される。

初回接続時は**Install Now**ボタンを押下しAspera Connectをインストールする。

The screenshot shows a Windows Internet Explorer browser window displaying the Aspera Connect server interface. The address bar shows the URL `http://133.39.224.17/aspera/user/`. The page header includes the Aspera Connect logo and the text "aspera connect server" and "Welcome ishikawa". A central message states "This site requires Aspera Connect." with a prominent "Install Now" button and a link to "click here to download the installer.". Below this, the IP address "133.39.224.17" is displayed. A toolbar contains buttons for "Download", "Upload File", "Upload Folder", "Delete", and "New Folder". A file listing table is shown with columns for "Name", "Size", and "Last Modified".

	Name	Size	Last Modified
<input type="checkbox"/>	backup		24-Feb-2012 14:15
<input type="checkbox"/>	ruby-1.8.7-p352		22-Feb-2012 16:16
<input type="checkbox"/>	src		22-Feb-2012 16:11
<input type="checkbox"/>	uqe-test		27-Feb-2012 13:39
<input type="checkbox"/>	a.png	33KB	09-Mar-2012 16:19
<input type="checkbox"/>	aspera.png	159KB	10-Mar-2012 16:04

# スパコンへのデータ転送⑤



## Aspera Connectを使用する(Web UI)

- ③ Aspera Connectインストール後、**Upload File**ボタンを押下し、転送したいファイルを選択するとスパコンへデータが転送される。

	Name	Size	Last Modified
<input type="checkbox"/>	backup		24-Feb-2012 14:15
<input type="checkbox"/>	ruby-1.8.7-p352		22-Feb-2012 16:16
<input type="checkbox"/>	src		22-Feb-2012 16:11
<input type="checkbox"/>	uqe-test		27-Feb-2012 13:39
<input type="checkbox"/>	a.png	33KB	09-Mar-2012 16:19
<input type="checkbox"/>	aspera.png	159KB	10-Mar-2012 16:04
<input type="checkbox"/>	b.png	37KB	09-Mar-2012 16:20
<input type="checkbox"/>	c.png	49KB	09-Mar-2012 16:20
<input type="checkbox"/>	d.png	46KB	09-Mar-2012 16:20

# スパコンへのデータ転送⑥



## Aspera Connectを使用する(コマンドプロンプト)

- ①Web UIと同様の手順でAspera Connectをインストールする。  
コマンドプロンプトを開き、

```
“(Aspera Connectのインストールパス)¥bin¥ascp.exe” -P 33001  
(転送するデータのパス) (ユーザ名)@ascp.ddbj.nig.ac.jp:(ホームディレク  
トリ以下の転送先のパス)
```

と入力する。パスワードを入力すると転送が開始される。

<実行例> (コマンドプロンプト) ※Windows7の場合

```
C:¥Users¥lect01>“C:¥Users¥lect01¥AppData¥Roaming¥Aspera¥Aspera  
Connect¥bin¥ascp.exe” -P 33001 C:¥Users¥lect01¥Desktop¥test.txt  
lect01@ascp.ddbj.nig.ac.jp:test.txt ←/home/lect01/test.txtにデータ転送  
lect01@ascp.ddbj.nig.ac.jp's password:  
test.txt 100% 0 00:00
```

# スパコンへのデータ転送⑦



## Aspera Connectを使用する(ターミナル)

- ①Web UIと同様の手順でAspera Connectをインストールする。  
ターミナルを開き、

```
“(Aspera Connectのインストールパス)/Resources/ascp”  
-P 33001 (転送するデータパス) (ユーザ名)@ascp.ddbj.nig.ac.jp:  
ホームディレクトリ以下の転送先のパス)
```

と入力する。パスワードを入力すると転送が開始される。

### <実行例> (ターミナル)

```
lect% ./Library/Application Support/Aspera Connect.app/Contents/Resources/ascp -P 33001 ./test.txt ¥  
lect01@ascp.ddbj.nig.ac.jp:test.txt ← /home/lect01/test.txtにデータ転送  
lect01@ascp.ddbj.nig.ac.jp's password:  
test.txt 100% 0 00:00
```

# スパコンへのデータ転送



## Aspera Connectのオプション

以下にAspera転送のオプションを示す。WebUI使用時はAspera ConnectのPreferencesから設定可能である。

-T

データの暗号化を行わない。転送速度は向上するが、セキュリティは低下する。

-l <転送速度上限>

データ転送速度の上限を設定する。デフォルトは10M(b/s)となっている。回線速度を上回る転送速度を設定した場合、パケットロスが大量に発生し、転送効率が大幅に低下するため適切な転送速度を設定する必要がある。上限は1G(b/s)。

<例>

```
ascp -l 100M -P 33001 [ユーザ名]@ascp.ddbj.nig.ac.jp:[転送先パス] [転送するデータ]
```



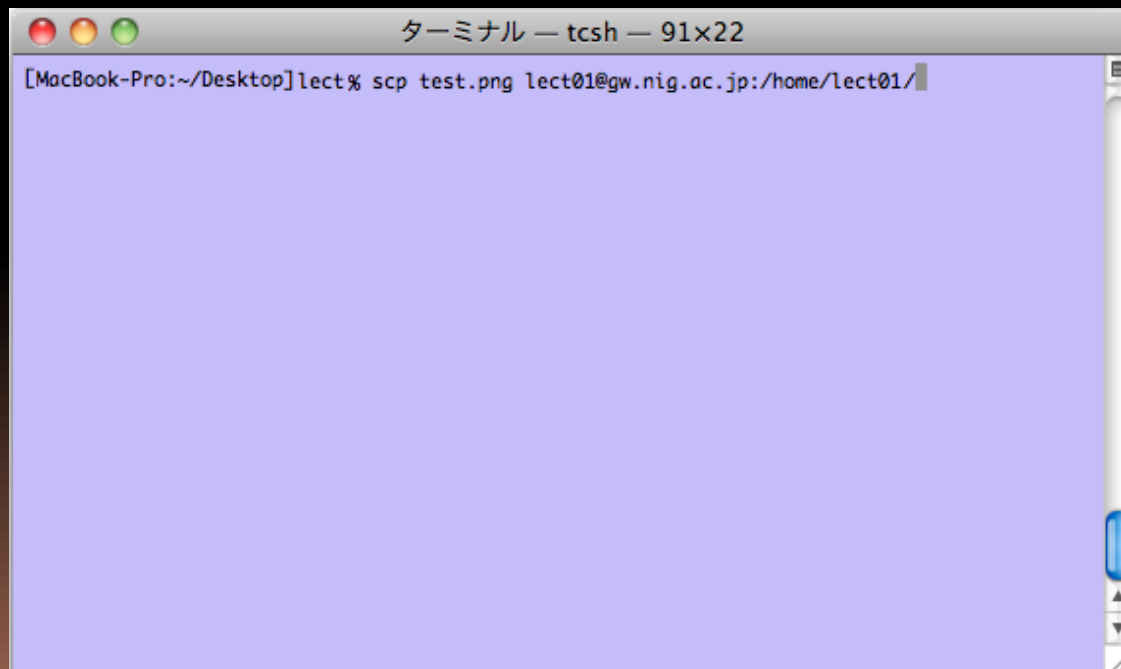
# スパコンへのデータ転送⑧



## ターミナルを使用する

①ターミナルを実行し、

`scp` (転送するデータのパス) (ユーザ名)`@gw.ddbj.nig.ac.jp:`  
(データを置くパス) と入力する。パスワードを入力するとデータが指定したパスに転送される。



```
ターミナル — tcsh — 91x22
[MacBook-Pro:~/Desktop]lect% scp test.png lect01@gw.nig.ac.jp:/home/lect01/
```

# ユーザへのお願い事項

- パスワードの管理を厳重にお願いします。
- 自分のアカウントを他人に貸し出さないで下さい。
- パスワード漏えいの疑いがある場合は、早急に最終ページ記載のスパコン管理チームまでご連絡下さい。

遺伝研スパコン管理チーム

Mail : [sc-info@nig.ac.jp](mailto:sc-info@nig.ac.jp)

Tel : 055-981-9461

# 問い合わせ先

- 不明点またはご意見等があれば下記にお問い合わせ下さい

遺伝研スパコン管理チーム

Mail : [sc-info@nig.ac.jp](mailto:sc-info@nig.ac.jp)

Tel : 055-981-9461

<http://sc.ddbj.nig.ac.jp/>

# 変更履歴

変更日付	変更内容
2013/11/25	新規作成