2012/6/26

スパコン利用者説明会
-ＵＧＥ概説-

Univa Grid Engine(UGE)とは

- グリッドコンピューティングシステムを構築するソフトウェア。バッチジョブシステムとして機能する
- Sun Grid Engine6.2U5(オープンソース版としては最後のバージョン)から派生した商用製品。開発にはGrid Engineの主要な開発メンバーが参加している
- ジョブ投入時のコマンド等はSGEと同じ

ＵＧＥ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

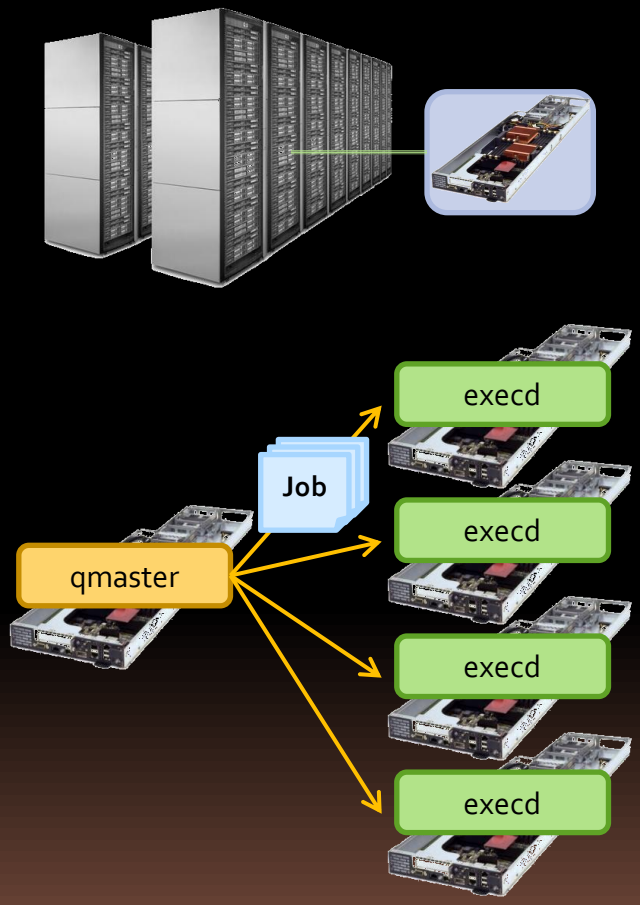
ＵＧＥを利用するうえでの注意点

- ジョブの並列化などは行わない
- 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

スパコン使用方法(イメージ)

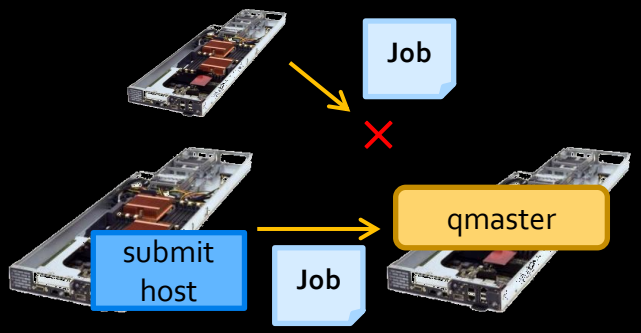
①ゲートウェイノード(gw.ddbj.nig.ac.jp)にログインする
②qloginを実行しインタラクティブノードにログインする
③qloginしたホストからジョブをUGEに投入する
④UGEは負荷の低いノードでジョブを実行する
⑤ジョブ実行結果をlustreのホームディレクトリに出力する
⑥ジョブ実行結果を確認する

基本用語(概念)1

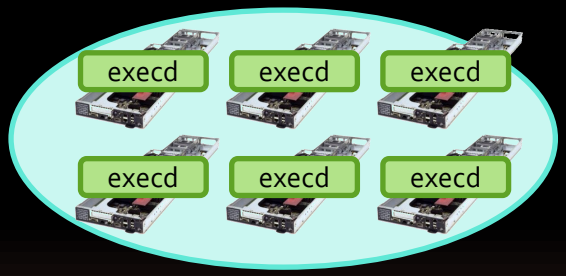


- ホスト（ノード）
  物理的に存在する計算機
- マスターホスト
  UGEのマスターデーモン(qmaster)が動作するホスト
  マスターデーモンはUGEを統括するデーモンで、ジョブの受付、スケジューリング、実行ホストへの配送、回収などを行う
- 実行ホスト
  UGEの実行デーモン(execd)が動作するホスト
  実行デーモンはマスターデーモンからのジョブ実行の指示を受け、ジョブを実行する

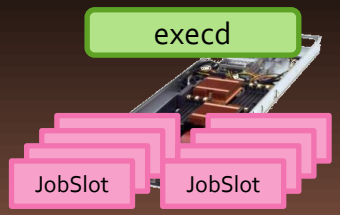
基本用語(概念)2



- 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

２つのＵＧＥ環境

- 本システムには、以下の２つのＵＧＥ環境がある。
- 使用可能な環境設定はログイン時に行われるため、利用者は設定作業を意識する必要はない

ＤＤＢＪ業務用ＵＧＥ環境
　ＤＤＢＪ業務用アカウントで使用できるＵＧＥ環境
　　SGE_ROOT=/home/geadmin/UGES
　　SGE_CELL=uges

研究用UGE環境
　一般研究用アカウントで使用できるＵＧＥ環境
　　SGE_ROOT=/home/geadmin/UGER
　　SGE_CELL=uger

キューの種類(研究用 6/26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に使用
login.q	128	ー	ジョブの投入を行うために使用

キューの種類(業務用 6/26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
month_hdd.q	448	62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
debug.q	32	1日	ジョブの動作確認をする場合に使用
login.q	64	ー	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
(研究用・DDBJ業務用共通)

```
$ 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
以下に例を示す（ファイル名は“test.sh”とする）

```
#!/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
/lustre1/home/ddbjuser
t165
2012年  3月 21日 水曜日 11:15:01 JST
2012年  3月 21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て表示される

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
ddbjuser 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 ddbjuser
ddbjuser has registered the job 327 for deletion
ddbjuser 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(研究用)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(研究用)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(研究用)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(研究用)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

キューの使い分け方法(業務用)

リソースを指定しない

```
$ qsub test.sh
```

month_hdd.qが使われる

“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

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

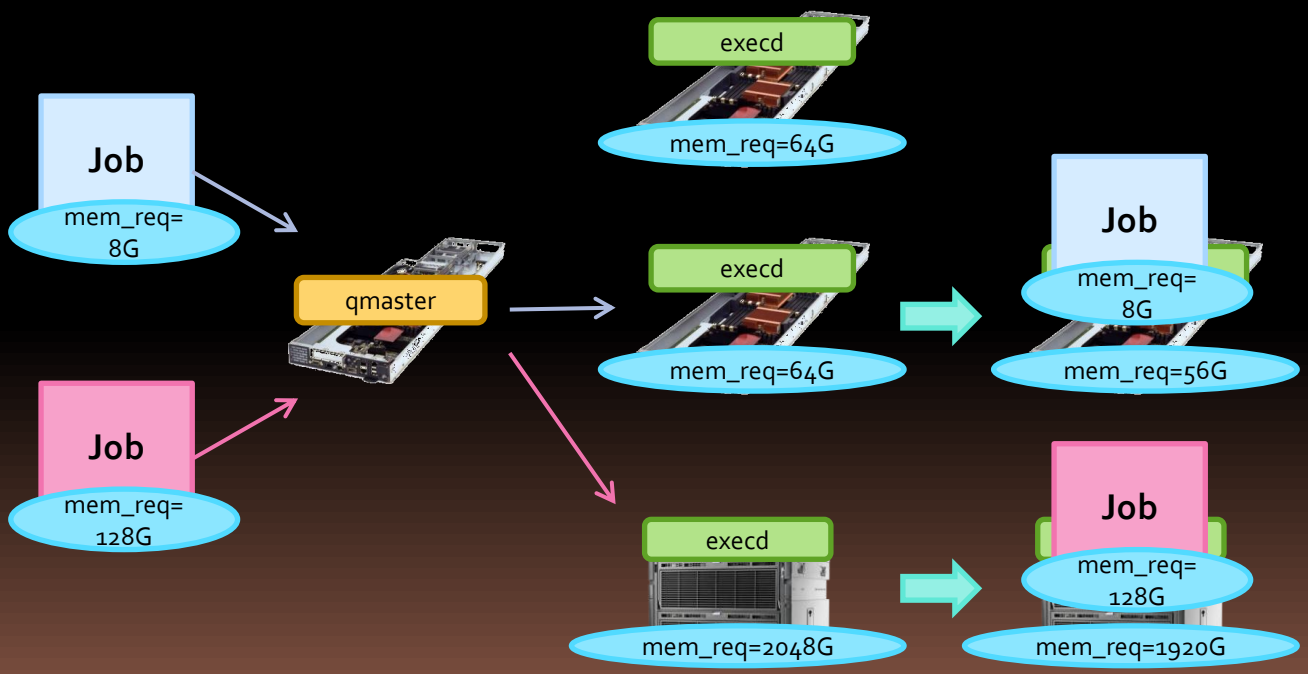
```
$ qsub -l s_vmem=8G -l mem_req=8G test.sh
```

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値を宣言する。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

アレイジョブ1

ジョブをアレイジョブとして投入すると、同一のジョブに
異なるパラメータを与えて繰り返し実行できる
qsubの“-t”オプションを用いるとアレイジョブを投入できる

```
$ cat arraytest.sh
#!/bin/sh
#$ -S /bin/sh

echo ---
echo JOB_ID: ${JOB_ID}
echo SGE_TASK_ID: ${SGE_TASK_ID}
echo SGE_TASK_FIRST: ${SGE_TASK_FIRST}
echo SGE_TASK_LAST: ${SGE_TASK_LAST}
echo SGE_TASK_STEPSIZE: ${SGE_TASK_STEPSIZE}
echo ---

$ qsub -t 1-6:2 arraytest.sh
Your job-array 1031.1-6:2 ("arraytest.sh") has been submitted
$ qstat
job-ID  prior   name       user         state submit/start at     queue                          slots ja-
task-ID
-----------------------------------------------------------------------------------------------------------
------
   1031 0.50000 arraytest. ddbjuser     r     03/19/2012 00:43:13 week_hdd.q@t168i                1 1
   1031 0.50000 arraytest. ddbjuser     r     03/19/2012 00:43:13 week_hdd.q@t168i                1 3
   1031 0.50000 arraytest. ddbjuser     r     03/19/2012 00:43:13 week_hdd.q@t178i                1 5
```

アレイジョブ2

```
$ ls arraytest.sh.o1031.*
arraytest.sh.o1031.1  arraytest.sh.o1031.3  arraytest.sh.o1031.5
$ cat arraytest.sh.o1031.1
---
JOB_ID: 1031
SGE_TASK_ID: 1
SGE_TASK_FIRST: 1
SGE_TASK_LAST: 6
SGE_TASK_STEPSIZE: 2
---
$ cat arraytest.sh.o1031.5
---
JOB_ID: 1031
SGE_TASK_ID: 5
SGE_TASK_FIRST: 1
SGE_TASK_LAST: 10
SGE_TASK_STEPSIZE: 2
---
```

UGEが過負荷となることを防ぐため、1ユーザあたりの投入可能なジョブ数には上限がある。上限を上回るジョブを投入しようとするとエラーになってジョブを投入できない
本システムでの1ユーザあたりのジョブ投入数上限は5000

アレイジョブとしてジョブを投入すると、UGEに与える負荷を軽減できる。
5000ジョブのアレイジョブを投入すれば、
5000 * SGE_TASK_ID分のジョブを実行できる
SGE_TASK_IDの上限は75000

MPIジョブ1

MPIジョブを投入するシェルスクリプトの例を以下に示す

```
$ cat mpitest.sh
#!/bin/sh

#$ -S /bin/sh
#$ -pe mpi 2-24
#$ -cwd

/usr/local/bin/mpirun -np $NSLOTS ¥
                      -machinefile $TMPDIR/machines ¥
                      ./mpitest
```

`-pe` <MPI実行環境名> <最小並列数>-<最大並列数>

MPI実行環境（後述）、最小並列数、最大並列数を指定する

`$NSLOTS`

キューの空き状況に応じて、上記の[最小並列数]～[最大並列数]から自動決定した値が設定される

`-machinefile $TMPDIR/machines`

ファイル$TMPDIR/machinesは、UGEが自動生成する

MPIジョブ2

MPIジョブをUGEに投入する

```
$ qsub mpitest.sh
Your job 1292 ("mpitest.sh") has been submitted
$ qstat
job-ID  prior   name       user         state submit/start at     queue
slots ja-task-ID
-----------------------------------------------------------------------------
-----------------------------
1292 0.50000 mpitest.sh ddbjuser     r     03/19/2012 20:55:24 week_hdd.q@t303i
24
$ cat mpitest.sh.o1292
Hellow World from Process 0 of 24 running on t303
Hellow World from Process 1 of 24 running on t290
(※中略※)
Hellow World from Process 19 of 24 running on t311
```

主なMPI実行環境：
- mpi：　並列ジョブを可能な限り多くのホストを利用して実行する
- mpi-fillup：並列ジョブを可能な限り同一のホストで実行する

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
```

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SE
Mail：sc-info@nig.ac.jp
居室：w202
内線：9461

http://www.ddbj.nig.ac.jp/system/supercom/supercom-intro.html

変更履歴

変更日付	変更内容
2012/03/21	新規作成
2012/05/10	「キューの種類」のジョブスロット数を現状に合わせて修正、def_slot使用時の注意事項を追記
2012/06/18	month系キューの実行時間上限を31日から62日に変更したことに伴い「キューの種類」の内容を修正
2012/06/26	キュー構成を見直したことに伴い、「キューの種類」の内容を修正
	業務用UGE環境のキューが減少したことにより「キューの使い分け方法（業務用）」が変わったため内容を修正