HPC SOLUTIONS

TORQUE　利用マニュアル

Ver.1.1

2011 年 8 月 8 日

株式会社 HPC-ソリューションズ

1 改訂履歴

版	改訂日	改訂理由	改訂内容	改訂者
1.0	2011 年 8 月 4 日	初版作成		吉岡　保
1.1	2011 年 8 月 8 日	2 版作成	実行方法追加	吉岡　保

2 TORQUE の概要

TORQUE（Tera-scale Open-source Resource and QUEue manager）とは、
Cluster Resource 社によって、Open PBS をもとに開発されているフリーのパッケージであり、現在（2011 年 7 月）のバージョンは 3.0.2 となっている。
Open PBS は 1998 年に開発が終了しており、Open PBS へいくつかの改良を加えて開発が行われているのが、TORQUE である。
そのため、基本構成（コマンドやコンポーネント構成、特徴等）は Open PBS（PBS）と同様であるところが多くある。
以下に Open PBS に追加された機能と特徴について示す。

3 TORQUE の運用イメージ

ヘッドノード：psi がジョブのスケジューリングおよびサーバーとして動作します。
psi001~psi009 までの 12 コア x9 台=108 コアを自動的に割り当てます。
・ヘッドノードに登録された計算ノードにジョブをサブミットします。
・ヘッドノードはデフォルトではジョブを実行しないよう設定しております。

4 TORQUE の構成

TORQUE の構成は以下の 3 つのデーモンから構成されている.
ディレクトリは/var/spool/torque をホームディレクトリとしてます。

4.1 ・Job Server

デーモン：pbs_serv

Job Server の主な機能は, バッチジョブの生成・受理, ジョブの修正, システム障害に対するジョブの保護, ジョブの実行（実際には下記の Job Executor へジョブの実行を要求する）である. Job Server は一つもしくは, 複数の queue（キュー）を管理し, ヘッドノードで稼働している.

4.2 ・Job Scheduler

デーモン：pbs_sched

Job Scheduler は，どのジョブをどの計算ノードで，いつ実行させるかといったポリシーの管理を含むデーモンであり，ヘッドノードで稼働している．TORQUE のスケジューラの基本ポリシーは FIFO（First In First Out）スケジューラとなっている．

4.3 ・Job Executor

デーモン：pbs_mom

Job Executor は，実際にジョブを実行するためのデーモンであり，計算ノードで稼働しています。

※ 通常、Server ノードではジョブを実行しない設定となっておりますので
デーモンは動作しておりません。

5 *TORQUE の主なコマンド*

TORQUE を利用してジョブの実行で利用する主なコマンドは以下のものとなります。

サーバ・クライアントコマンド

Command	Description
pbsnodes	実行可能なノードの一覧を表示する。
qdel	ジョブを削除する
qhold	ジョブを実行不可にする
qmgr	キューを管理する。
qrls	ジョブを実行可にする
qrun	キューをスタートする。start a batch job
qsub	ジョブを投入する
qstat	キュー，ジョブの状態を表示する
qterm	pbs server デーモンをシャットダウンする。
tracejob	実行されたジョブのサービスログファイルメッセージを出力する。

6 ジョブの実行

6.1 ジョブの投入

バッチジョブのキューへの投入は qsub コマンドにより行います。
以下に qsub コマンドの使用方法を示します。
ジョブを投入する方法としては, 標準入力とシェルスクリプトを利用する二通りの方法がありますが、本編ではシェルスクリプトを利用する方法を記述します。

以下のようなシェルスクリプト（test.sh）を記述しジョブの実行をします。

```
[hpcs@psi ~]$ cat test.sh
#!/bin/sh
#PBS -l nodes=1:ppn=4
#PBS -q default
#PBS -N pbs-test
#PBS -j oe

cd $PBS_O_WORKDIR

./a.out      →  実行ジョブ
```

上記のシェルスクリプトにおいて「#PBS」というのは, コメントアウトではなく, TORQUE へのコマンドオプションになります。
詳細なコマンドオプションは”6.2 qsub のオプション”に記載します。

サブミットする際は下記の様に qsub コマンドに続きシェルスクリプトを入力します。

```
[hpcs@psi~]$ qsub test.sh
93.psi
```

正常にジョブがキューイングされると{JOB_ID}.sandy サーバー名)の
メッセージが出力されます。

6.2 qsub のオプション

下記は qsub の主なオプションとなります。
"-q"オプションで queueName を設定しない場合 default の queue で実行されます。

オプション	指定名	説明
-N	jobName	ジョブの名前
-q	queueName	キューの指定
-o filepath	outFile Name	標準出力をファイルに保存
-e filepath	errorFile Name	エラーの出力をファイルに保存
-j	eo	エラーを標準出力にまとめて保存
-l	ppn	1 つのジョブの 1 つのノード（グループ）で必要とする CPU 数
	nodes	1 つのジョブで必要とするノード数(グループ)
-m	a	ジョブがエラー終了した場合にメールを送る
	b	ジョブのスタート時点でメールを送る
	e	ジョブの終了時点でメールを送る
	abe	上記の 3 つを指定する場合は続けて記述

7 ジョブの状態を表示

投入したジョブの状態を表示するには，「qstat」コマンドを使用します。

7.1 qstat 実行例

qstat コマンドを使用すると現在キューイングされている
job が表示されます。

JobID、JobName、実行ユーザーなどのステータスが表示されます。

7.1.1 qstat 項目名

- JobID : ジョブ ID と実行 Server 名
- Name : スクリプト名もしくは指定のジョブネーム
- User : ユーザー名
- Time : ジョブの CPU 使用時間
- Use : ジョブの現在のステータス

-------ステータス-------

R = ラン 実行中
Q = キューイング キュー待機状態
H = ホールド 保留状態
E = イグジッティング 実行終了
T = ジョブは移行中
W = WAIT 待機状態

```
[hpcs@psi. ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
17.psi                    test.sh          hpcs            00:00:00 R default
18.psi                    pbs-test.sh      hpcs            00:00:00 R default
19. psi                   test3.sh         hpcs            00:00:00 Q default
```

7.1.2　詳細な qstat のステータス

-a オプションを使用する事で更に詳細なシステムの全てのジョブの内容が表示されます。

```
[hpcs@psi ~]$ qstat -a
psi:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname    SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------- ------ ----- ------ ------ ----- - -----
17.psi               hpcs     default  test        9818   1       1   --     --   R 00:00
18.psi               hpcs     default  test        9919   1       1   --     --   R 00:00
                                      :
```

表示の内容

- Job ID　：　ジョブ ID と実行 Server 名
- Username　：　ユーザー名
- Queue　：　キューネーム
- jobname　：　スクリプト名もしくは指定のジョブネーム
- SessID　：　セッション ID
- NDS　：　要求されたノードの数
- TSK　：　同時タスク（または CPU）の数
- Req'd Memory：　要求されたメモリの量
- Req'd Time　：　要求された経過時間
- S　：　ジョブの現在のステータス
- Elap Time　：　現在のジョブ状態の経過時間

※Qstat はジョブがキューイングされていない状態では何も表示されません。

7.2 主な qstat のオプション

オプション	説明
-q	システムの全てのキューの状況を表示
-a	システムの全てのジョブの状況を表示
-s	全てのジョブをステータスコメント付きで表示
-r	実行中の全てのジョブを表示
-B	PBS Server のサマリ情報を表示する
-Q	全てのキューのリミット値を表示する

-au {userid}	指定したユーザのジョブを表示する
-f {JobID}	指定したジョブの詳細を表示する
-Qf {queue}	指定したキューの詳細を表示する

8 *Queue の状態を見る*

現在設定されている詳細な Queue の状態またはサーバーの状態を確認するためには qstat コマンドのオプションに「-B -f」をつけます。

```
 [hpcs@psi ~]$ qstat   -B -f
Server: psi
      server_state = Active
      server_host = psi.issp.u-tokyo.ac.jp
      scheduling = True
      total_jobs = 0
      state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0 Begun:0
      default_queue = default
      log_events = 511
      mail_from = adm
      query_other_jobs = True
      resources_assigned.ncpus = 1
      defau;t_chunk.ncpus = 1
      resources_assigned.ncpus = 0
      resources_assigned.nodect = 0
      scheduler_iteration = 600
      FLicenses = 48
      Resv_enable = True
      Node_fail_requeue=310
                                  :
```

9 ジョブの削除

ジョブの削除には, 「qdel 」コマンドを使用して, Job ID を指定します。

書式 : [hpcs@psi～]$ qdel [JobID]

9.1 実行例

・Job ID は上記の qstat で取得した Job ID である. 以下に実行例を示す.

```
[hpcs@psi ~]$ qstat -a

psi:
                                                                  Req'd  Req'd   Elap
Job ID                  Username Queue    Jobname    SessID NDS   TSK Memory Time   S Time
----------------------- -------- -------- ---------- ------ ----- ------ ------ ----- - -----
17.psi                  hpcs     default  test        9818   1      1   --     --    R 00:00
18.psi                  hpcs     default  test        9919   1      1   --     --    R 00:00
```

・ JobID18 を削除して qstat コマンドを実行ます。

```
[hpcs@psi ~]$ qdel 18
[hpcs@psi ~]$ qstat –a

psi:
                                                                  Req'd  Req'd   Elap
Job ID                  Username Queue    Jobname    SessID NDS   TSK Memory Time   S Time
----------------------- -------- -------- ---------- ------ ----- ------ ------ ----- - -----
17.psi                  hpcs     default  test        9818   1      1   --     --    R 00:00
```

qstat –a の出力から JobID18 が削除されていることを確認します。

10 *実行したジョブをトレースする*

Tracejob コマンドで実行したジョブをトレースすることができます。

[実行例]

・JobID 17 を実行中、または実行後に

```
[hpcs@psi ~]$ tracejob 17

Job: 17.psi

07/26/2011 21:17:58  S     Job Queued at request of hpcs@ psi, owner =
                           hpcs@ psi, job name = test.sh, queue = default
07/26/2011 21:17:58  S     Job Modified at request of Scheduler@ psi
07/26/2011 21:17:58  L     Not enough of the right type of nodes available
07/26/2011 21:17:58  S     enqueuing into default, state 1 hop 1
07/26/2011 21:21:20  S     Job Modified at request of Scheduler@ psi
07/26/2011 21:21:20  L     Job Run
07/26/2011 21:21:20  S     Job Run at request of Scheduler@psi
07/26/2011 21:24:37  S     Exit_status=0 resources_used.cput=00:00:42
                           resources_used.mem=102256kb
resources_used.vmem=32367300kb
                           resources_used.walltime=00:00:43
07/26/2011 21:24:37  S     dequeuing from test, state 5
```

[tracejob の主なオプション]

-p	path to PBS_SERVER_HOME
-w	number of columns of your terminal
-n	number of days in the past to look for job(s) [default 1]
-f	filter out types of log entries, multiple -f's can be specified error, system, admin, job, job_usage, security, sched, debug, debug2, or absolute numberic equiv
-z	toggle filtering excessive messages
-c	what message count is considered excessive
-a	don't use accounting log files
-s	don't use server log files
-l	don't use scheduler log files
-m	don't use mom log files
-v	verbose mode - show more error messages default prefix path = /var/spool/torque

11 Queue の管理

Queue 設定の管理は qmgr コマンドを使用します。

11.1 現在の Queue の設定を出力する

現在、クラスタに設定されている Queue を qmgr コマンドを使用して出力します。

```
#
# Create queues and set their attributes.
#
#
# Create and define queue large
#
create queue large
set queue large queue_type = Execution
set queue large max_running = 1
set queue large resources_max.ncpus = 48
set queue large resources_max.nodect = 4
set queue large resources_min.ncpus = 1
set queue large resources_min.nodect = 1
set queue large resources_default.ncpus = 48
set queue large resources_default.nodect = 4
set queue small resources_default.walltime = 01:00:00
set queue small max_user_run = 1
set queue large enabled = True
set queue large started = True
#
# Create and define queue small
#
create queue small
set queue small queue_type = Execution
set queue small max_running = 9
set queue small resources_max.ncpus = 12
set queue small resources_max.nodect = 1
set queue small resources_min.ncpus = 1
```

```
set queue small resources_min.nodect = 1
set queue small resources_default.ncpus = 12
set queue small resources_default.nodect = 1
set queue small resources_default.walltime = 00:30:00
set queue small max_user_run = 9
set queue small enabled = True
set queue small started = True
#
# Create and define queue default
#
create queue default
set queue default queue_type = Execution
set queue default max_running = 9
set queue default resources_max.ncpus = 12
set queue default resources_max.nodect = 1
set queue default resources_min.ncpus = 1
set queue default resources_min.nodect = 1
set queue default resources_default.ncpus = 12
set queue default resources_default.nodect = 1
set queue default resources_default.walltime = 00:30:00
set queue default max_user_run = 9
set queue default enabled = True
set queue default started = True
#
# Create and define queue middle
#
create queue middle
set queue middle queue_type = Execution
set queue middle max_running = 4
set queue middle resources_max.ncpus = 24
set queue middle resources_max.nodect = 2
set queue middle resources_min.ncpus = 1
set queue middle resources_min.nodect = 1
set queue middle resources_default.ncpus = 24
set queue middle resources_default.nodect = 2
set queue default resources_default.walltime = 00:30:00
```

```
set queue default max_user_run = 4
set queue middle enabled = True
set queue middle started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_hosts = psi
set server managers = hpcs@psi.issp.u-tokyo.ac.jp
set server operators = hpcs@psi.issp.u-tokyo.ac.jp
set server default_queue = default
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
set server node_check_rate = 60
set server tcp_timeout = 6
set server mom_job_sync = True
set server auto_node_np = True
set server next_job_number = 26
```

11.1.1 qmgr –c “p s” の出力の見方

1,"default_queue = default"デフォルトの Queue が"default"になっております。
何もご指定が無い場合はデフォルトの queue に投入されます。
2,"scheduling = True"スケジューリングが有効になっているか確認します。

11.1.2 Qmgr の主な使い方

・qmgr -c "delete queue QNAME"
queue（queue 名が QNAME)を削除する

・qmgr -c "create queue QNAME"
queue（queue 名が QNAME)を作成する

・qmgr -c "set queue QNAME max_running = 10"
queue（queue 名が QNAME)の実行可能 job 数を 10 にする

・qmgr -c "set queue QNAME max_user_run = 3"
queue（queue 名が QNAME)の 1 ユーザが実行可能な job 数を 3 にする

・qmgr -c "unset queue QNAME max_user_run"
queue（queue 名が QNAME)の 1 ユーザが実行可能な job 数を設定なしにする

・set queue [QNAME] resources_default.walltime = 01:00:00
キューに実行時間制限を 1 時間とする。

11.2　　サンプル 1

11.2.1　QueueName が workq で無制限のキューを作成する方法

- 実行時間の制限なし
- 実行メモリーの制限なし
- 実行ノードの制限なし
- 実行ユーザーの制限なし
- 実行ノード内 CPU 数のデフォルト値なし
- デフォルトで実行される Queue に設定する。

```
qmgr -c "create queue workq"
qmgr -c "set queue workq queue_type = Execution"
qmgr -c "set queue workq enabled = True"
qmgr -c "set queue workqt started = True"
qmgr -c "set server default_queue = workq"
```

11.2.2　QueueName が nodeq で実行ノードに制限設定があるキュー

※1 ノード 2CPU あるノードを 4 台、合計 8 並列以上リソースがある場合に設定します。

- 実行時間の制限なし
- 実行メモリーの制限なし
- 実行ノードの制限数： 4
- 実行プロセッサー数制限：8
- 実行ジョブ数制限：2

・ 実行ユーザーの制限なし

```
qmgr –c “create queue nodeq”
qmgr –c “set queue nodeq queue_type = Execution”
qmgr –c “set queue nodeq enabled = True”
qmgr –c “set queue nodeq started = True”
qmgr –c “set queue nodeq max_running = 2”
qmgr –c “set queue nodeq resources_max.ncpus = 8”
qmgr –c “set queue nodeq resources_max.nodect = 4”
```

11.2.3 QueueName が userq でユーザーの実行数に制限設定があるキュー

・ 実行時間の制限：なし
・ 実行メモリーの制限：なし
・ 実行ノードの制限数：なし
・ 実行プロセッサー数制限：なし
・ 実行ジョブ数制限：なし
・ 実行ユーザーの制限：1 ユーザー3 ジョブまでの制限
・ プライオリティー設定なし

```
qmgr -c "create queue userq”
qmgr -c "set queue userq queue_type = Execution"
qmgr -c "set queue userq enabled = True"
qmgr -c "set queue userq started = True"
qmgr -c "set server userq_queue = default"
qmgr -c "set server userq_other_jobs = True"
qmgr -c "set queue userq max_user_run = 3”
```

12 *ノードの状態を見る*

ノードの状態を見るためには pbsnodes コマンドを使用します。
本コマンドを使用すると、ノードの状態を照会したり、ノードを停止状態、フリー状態、またはオフライン状態としてマーキングすることができます。

コマンドの使用方法は以下のとおりです。

pbsnodes [-a | -l | -s]　[-c node] [-d node] [-o node] [-r node] [node1 node2 …]

[pbsnodes の主なオプション]

Option	説明
なし	コマンドに使用する構文を出力
node1　node2	node1、node2 のノード状態を出力
-a	すべてのノードとそのすべての属性をリスト表示
-c nodes	リストされたノードの OFFLINE または DOWN 状態を解除。リストされたノードは、ジョブへの割り当てが可能な”フリー”状態になります。
-d nodes	オペランドとして指定したノードは DOWN としてマークされ、ジョブの実行には使用できなくなります。停止していることが分かっている全てのノードを、このコマンド行の引数として指定することが重要です。これは、指定されていないノードは稼動状態とみなされ、その前に DOWN とマークされた場合でも、稼動状態として表示されてしまうためです。つまり”pbsnodes –d”と指定すると、すべてのノードがフリーとしてマークされます。
-l	何らかのマークが設定されたすべてのノードをリスト表示
-o nodes	リストされたノードを現在使用中だとしても、OFFLINE とマーキングする。ノードが稼動中か停止中かをチェックし、停止中のノードリストによって pbsnodes を呼び出す自動化スクリプトは OFFLINE とマークされたノードの状態を変更しません。 つまり、アドミニストレータはこのオプションを使用することで、自動化スクリプトを変更することなく、ノードを停止状態に維持できます。
-r nodes	リストされたノードの OFFLINE 状態を解除
-s	接続先の pbs_serv ホストを指定

[実行例]

pbsnodes コマンドで現在設定されているノードからノードの現在のステータスを表示します。

```
[hpcs@psi ~]$ pbsnodes -a
        psi001
             state = free
             np = 12
             ntype = cluster
```

```
        status                                                                                                  =
rectime=1311849510,varattr=,jobs=,state=free,netload=1479535015,gres=,loadave=0.00,ncpus=12,physmem=
24723912kb,availmem=32367300kb,totmem=33112512kb,idletime=0,nusers=1,nsessions=1,sessions=2125,un
ame=Linux psi001.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011
x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi002
        state = free
        np = 12
        ntype = cluster
        status                                                                                                  =
rectime=1311849538,varattr=,jobs=,state=free,netload=1480024179,gres=,loadave=0.00,ncpus=12,physmem=
24723912kb,availmem=32383732kb,totmem=33125896kb,idletime=0,nusers=0,nsessions=0,uname=Linux
psi002.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011 x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi003
        state = free
        np = 12
        ntype = cluster
        status                                                                                                  =
rectime=1311849553,varattr=,jobs=,state=free,netload=1488885071,gres=,loadave=0.00,ncpus=12,physmem=
24723912kb,availmem=32381008kb,totmem=33125896kb,idletime=0,nusers=1,nsessions=1,sessions=1974,un
ame=Linux psi003.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011
x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi004
        state = free
```

HPC SOLUTIONS

```
        np = 12
        ntype = cluster
        status                                                                                   =
rectime=1311849528,varattr=,jobs=,state=free,netload=1488644870,gres=,loadave=0.00,ncpus=12,physmem=
24723912kb,availmem=32391984kb,totmem=33125896kb,idletime=0,nusers=0,nsessions=0,uname=Linux
psi004.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011 x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi005
        state = free
        np = 12
        ntype = cluster
        status                                                                                   =
rectime=1311849553,varattr=,jobs=,state=free,netload=730241457,gres=,loadave=0.00,ncpus=12,physmem=2
4723912kb,availmem=32396464kb,totmem=33125896kb,idletime=0,nusers=1,nsessions=1,sessions=1921,una
me=Linux psi005.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011
x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi006
        state = free
        np = 12
        ntype = cluster
        status                                                                                   =
rectime=1311849530,varattr=,jobs=,state=free,netload=729882627,gres=,loadave=0.02,ncpus=12,physmem=2
4723912kb,availmem=32394488kb,totmem=33125896kb,idletime=0,nusers=0,nsessions=0,uname=Linux
psi006.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011 x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi007
```

```
        state = free
        np = 12
        ntype = cluster
        status                                                                                          =
rectime=1311849544,varattr=,jobs=,state=free,netload=744205881,gres=,loadave=0.00,ncpus=12,physmem=2
4723912kb,availmem=32393568kb,totmem=33125896kb,idletime=0,nusers=1,nsessions=1,sessions=1926,una
me=Linux   psi007.issp.u-tokyo.ac.jp   2.6.32-71.el6.x86_64   #1   SMP   Fri   May   20   03:51:51   BST   2011
x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi008
        state = free
        np = 12
        ntype = cluster
        status                                                                                          =
rectime=1311849534,varattr=,jobs=,state=free,netload=747223876,gres=,loadave=0.01,ncpus=12,physmem=2
4723912kb,availmem=32394612kb,totmem=33125896kb,idletime=0,nusers=0,nsessions=0,uname=Linux
psi008.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011 x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0

psi009
        state = free
        np = 12
        ntype = cluster
        status                                                                                          =
rectime=1311849518,varattr=,jobs=,state=free,netload=245780013,gres=,loadave=0.00,ncpus=12,physmem=2
4723912kb,availmem=32357576kb,totmem=33125896kb,idletime=0,nusers=0,nsessions=0,uname=Linux
psi009.issp.u-tokyo.ac.jp 2.6.32-71.el6.x86_64 #1 SMP Fri May 20 03:51:51 BST 2011 x86_64,opsys=linux
        mom_service_port = 15002
        mom_manager_port = 15003
        gpus = 0
```

※ pbsnodes の出力をみますと psi001~psi009 の計 9 ノードあり、” state”ステータスが”free”となっていますので全ノード使用できます。

13 ログファイル

TORQUE の各デーモンのログファイルはホームディレクトリ以下に作られます。
ホームディレクトリ：/var/spool/torque

pbs_mom ログ：/var/spool/torque/mom_logs
pbs_sched ログ：/var/spool/torque/sched_logs
pbs_serv ログ：/var/spool/torque/server_logs

14 実行例

MPI プログラムを使用してサンプルプログラムを実行してみます。
各々のキューに合わせた投入用のシェルスクリプトを作成する必要があります。
（スクリプトの例を次項 15 で説明しています）
hpcs というユーザーがあり、/home/hpcs/test/以下に実行プログラム(IMB-MPI1)とシェルスクリプト(middle.sh)があるとします。
各ノードは HEX-CORE CPU が 2 基搭載されてますので、6 core x 2Processor x 1 ノード=12CPU(12core)の構成となってます。
計算ノード間の通信は 1GigaBits EtherNet で行います。

14.1 実行例

- 実行スクリプトを確認します。ここでは、middle キューで openmpi での 24 並列について説明します。

HEX-CORE CPU x 2 構成のノードが 9 台ですので、24 並列のジョブが最大で 4 本実行可能です。
”nodes=N” N の値にジョブが必要とするノード数(グループ数)を指定します。
次に”ppn=N” N の値に 1 ノード当たりで必要となる core 数を指定します。middle キューでは基本､排他的に 24 並列実行であるので､必要なノード数は 2 となり､nodes=2､ppn=12 の指定となります。

```
 [hpcs@psi test]$ cat middle.sh
#!/bin/sh
#PBS –l nodes=2:ppn=12
#PBS -j oe
#PBS –q middle
#PBS –N pbs-test

cd $PBS_O_WORKDIR
NPROCS=`wc -l < $PBS_NODEFILE`
cat  $ PBS_NODEFILE

mpirun –np $NPROCS –machinefile $PBS_NODEFILE ./a.out
```

・ 実行されるノード

実行されるノードは TORQUE 任せとなります。-machinefile に与えるホストファイルには TORQUE が割り当てたノードのホスト名が記述されています。

・ジョブをサブミットします。

```
[hpcs@psi test]$ qsub middle.sh
74.sandy
```

正常にジョブがキューイングされると{JOB_ID}.server(TORQUE サーバー名)のメッセージが出力されます。

・qstat で現在のジョブの状態をみます。

```
[hpcs@psi ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
74.psi                    pbs-test         hpcs                   00:00:00 R middle
```

・ジョブが終了しましたらアウトファイルに実行結果が書き込まれます。

```
[hpcs@psi ~]$ cat pbs-test.o74
psi001
psi001
 :
psi001
psi002
psi002
 :
psi002
#------------------------------------------------
#      Intel (R) MPI Benchmark Suite V3.2.2, MPI-1 part
#------------------------------------------------
# Date                  : Tue Jul 26 11:57:27 2011
# Machine               : x86_64
# System                : Linux
# Release               : 2.6.32-71.el6.x86_64
# Version               : #1 SMP Fri May 20 03:51:51 BST 2011
# MPI Version           : 2.1
```

```
# MPI Thread Environment: MPI_THREAD_SINGLE



# New default behavior from Version 3.2 on:

# the number of iterations per message size is cut down
# dynamically when a certain run time (per message size sample)
# is expected to be exceeded. Time limit is defined by variable
# "SECS_PER_SAMPLE" (=> IMB_settings.h)
# or through the flag => -time



# Calling sequence was:

# ./IMB-MPI1
                        :
#---------------------------------------------------------------------------
# Benchmarking Sendrecv
# #processes = 24
#---------------------------------------------------------------------------
                        :
#-------------------------------------------------
# Benchmarking Barrier
# #processes = 24
#-------------------------------------------------
  #repetitions    t_min[usec]    t_max[usec]    t_avg[usec]
             1000            51.18            51.21            51.19



# All processes entering MPI_Finalize
```

15 投入サンプルスクリプト集

15.1 シェルスクリプト

各キューに合わせた基本的な投入用のサンプルスクリプトを提示させて頂きます。
用途に合わせて改変してご利用下さい。

作成例）

```
[hpcs@psi ~] ~ $ vi middle.sh　：vi コマンド等のエディタで PBS 投入用スクリプトを作成。
```

15.2 サンプルスクリプト集

本説明中のスクリプトにおいてユーザーで変更する主なオプションは以下の赤字箇所になります。それ以外の箇所を変更された場合、動作しなくなることもありますので、なるべく指定箇所以外の変更はお奨め致しません。オプションの詳細は「章 6.2 qsub のオプション」を参照下さい。

- #PBS –j oe：標準出力、エラー出力を別々に指定
- #PBS –N test：ジョブ名を任意に指定
- <実行モジュール>：実行させるモジュールやプログラム等を指定。
- openmpi、mpich2 共に実行は基本同じ形式となっています。
- OMP_NUM_THREADS を指定する事で、openmp での投入/実行が可能です。

15.2.1 シングルジョブ実行スクリプト

【例：Small キューで a.out プログラムを実行】

```
#!/bin/sh
#PBS –l nodes=1:ppn=12
#PBS -j oe
#PBS –q small
#PBS –N test

cd $PBS_O_WORKDIR
NPROCS=`wc -l < $PBS_NODEFILE`
./a.out
```

15.2.2 MPI ジョブ実行スクリプト

【例：Small キュー】

```
#!/bin/sh
#PBS –l nodes=1:ppn=12
#PBS -j oe
#PBS –q small
#PBS –N test

cd $PBS_O_WORKDIR
NPROCS=`wc -l < $PBS_NODEFILE`
mpirun –np $NPROCS –machinefile $PBS_NOODEFILE <実行モジュール>
```

【例：middle キュー】

```
#!/bin/sh
#PBS –l nodes=2:ppn=12
#PBS -j oe
#PBS –q middle
#PBS –N test

cd $PBS_O_WORKDIR
NPROCS=`wc -l < $PBS_NODEFILE`
mpirun –np $NPROCS –machinefile $PBS_NOODEFILE <実行モジュール>
```

【例：large キュー】

```
#!/bin/sh
#PBS –l nodes=4:ppn=12
#PBS -j oe
#PBS –q large
#PBS –N test

cd $PBS_O_WORKDIR
NPROCS=`wc -l < $PBS_NODEFILE`
mpirun –np $NPROCS –machinefile $PBS_NOODEFILE <実行モジュール>
```

15.2.3　Hybrid ジョブ実行スクリプト

【例：large キュー、4node、4 スレッド/node】

```
#!/bin/sh
#PBS –l nodes=4:ppn=4
#PBS -j oe
#PBS –q large
#PBS –N test

export OMP_NUM_THREADS=4
cd $PBS_O_WORKDIR
mpirun –np 4 –npernode 1 <実行モジュール>
```