LSF・Lava ユーザーマニュアル

目次

1 IBM Spectrum LSF・Lava について

IBM Spectrum LSF は、ミッションクリティカルな計算・解析・シミュレーション等のアプリケーションのジョブスケジューリング処理を、高速・高スループットで運用可能にします。IBM Spectrum LSF を使用することにより、HPC 環境をインテリジェントなスケジューリングで実行することができます。業種や分野を問わず、既存ハードウェアリソースの使用率を最大化させます。

Lava は HPCCommunity.org のサイトからオープンソースのソフトウェアとしてリリースされていたジョブスケジューラーです。Lava は LSF をベースにしたソフトウェアで LSF とほぼ同様にシステムを運用することができます。

Lava は有償版の IBM Spectrum LSF と比較すると以下の制限があります。

・1 システム当りノード数の上限が 512 まで
・使用できるコマンドや機能が基本的なものに限る
・優先度の高いジョブが実行中のジョブを割り込んで実行することができない
・ジョブスケジューリングが LSF より簡易である
　複雑なスケジューリングを指定した場合は動作しないことがある
・Windows の未サポート
・LSF は多くの ISV ソフトウェアに対応しているが、Lava では動作しない場合がある

本マニュアルでは LSF と Lava のインストール内容、使用方法について概説します。LSF または Lava をご使用の際は本マニュアルをご覧下さい。

2　LSF インストール概要

(1) パッケージ

開発元が配布する IBM Spectrum LSF のメディアを使用します。

(2) インストールディレクトリ

`/usr/share/lsf`

クラスタ構成ではヘッドノードのディレクトリを NFS で共有します。

(3) 管理ユーザー

`lsfadmin`

(4) LSF デーモン

`/etc/init.d/lsf` … LSF クラスタの全ノードで起動します

OS 起動時に自動で `lsf` が起動します。

(5) 環境設定ファイル

LSF を使用するための環境設定は各ユーザーのホームディレクトリのファイルで行われています。tcsh をご使用の場合は `~/.cshrc` 、bash をご使用の場合は `~/.bashrc` ファイル内で `/home/.common` 以下に用意した LSF の環境設定スクリプトを実行します。

なお、標準設定では root は LSF の環境がセットされていないのでご注意下さい。root で LSF のコマンドを使用する際は以下コマンドで LSF 環境をセットして下さい。

tcsh の場合は以下コマンドを実行します。

```
# source␣/home/.common/00-lsf.csh
```

bash の場合は以下コマンドを実行します。

```
# .␣/home/.common/00-lsf.sh
```

3　Lava インストール概要

(1) パッケージ

Lava のソースコードは HPCCommunity.org のサイトで配布されていました。当社では `Lava-src-1.0.6.tar.gz` のソースファイルをビルドしています。

(2) インストールディレクトリ

`/usr/share/lava`

クラスタ構成ではヘッドノードのディレクトリを NFS で共有します。

(3) 管理ユーザー

`lavaadmin`

(4) Lava デーモン

`/etc/init.d/lava`

OS 起動時に自動で `lava` が起動します。

(5) 環境設定ファイル

Lava を使用するための環境設定は各ユーザーのホームディレクトリのファイルで行われています。tcsh をご使用の場合は `~/.cshrc` 、bash をご使用の場合は `~/.bashrc` ファイル内で `/home/.common` 以下に用意した Lava の環境設定スクリプトを実行します。

なお、標準設定では root は Lava の環境がセットされていないのでご注意下さい。root で Lava のコマンドを使用する際は以下コマンドで Lava 環境をセットして下さい。
tcsh の場合は以下コマンドを実行します。

```
# source␣/home/.common/00-lava.csh
```

bash の場合は以下コマンドを実行します。

```
# .␣/home/.common/00-lava.sh
```

4　コマンド例

本項では LSF・Lava で使用するコマンドを概説します。

(1) `lsload`

ノードの負荷情報を表示します。負荷情報はノードごとに、またはリソースごとに表示します。デフォルトではクラスタ内の全ノードの負荷情報を表示します。リソースの負荷情報は CPU とページング負荷に基づいて表示されます。

```
$ lsload
HOST_NAME      STATUS   r15s  r1m  r15m  ut  pg  ls  it  tmp    swp  mem
hpcs01.localhost  ok    0.0   0.0   0.0   0%  0.0  2  1 903G  3906M  31G
hpcs02.localhost  ok    0.0   0.0   0.0   0%  0.0  2  1 903G  3906M  31G
hpcs03.localhost  ok    0.0   0.0   0.0   0%  0.0  2  1 903G  3906M  31G
hpcs04.localhost  ok    0.0   0.0   0.0   0%  0.0  2  1 903G  3906M  31G
```

・STATUS：各ノードの状態を示します。
　ok ･･･ 正常
　busy ･･･ ノードの CPU 使用率が 100%近くである
　unavail ･･･ スケジューラーが正常に動作していない
　　OS 起動直後やマシンが起動していない場合にこのステータスになる場合があります
・r15s：直前の 15 秒間の CPU 使用率の平均
・r1m：直前の 1 分間の CPU 使用率の平均
・r15m：直前の 15 分間の CPU 使用率の平均
・ut：直前の 1 分間の CPU 使用率の平均
　0～100%の間で表示
・pg：直前の 1 分間にわたって指数平均をとったメモリページング率
・ls：現在のログインユーザの数
・it：現在までのアイドル時間[分]
・tmp：`/tmp` 内の空き領域のサイズ
・swp：使用可能なスワップスペースのサイズ
・mem：使用可能なメモリのサイズ

(2) bhosts

各ノードのジョブの動作を表示します。デフォルトでは、全ノードの情報(ノード名、ノード状態、ジョブスロット制限、およびジョブ状態統計)を表示します。

```
$ bhosts
HOST_NAME       STATUS   JL/U   MAX   NJOBS  RUN  SSUSP  USUSP   RSV
hpcs01.localhost   ok       -    16       0    0      0      0     0
hpcs02.localhost   ok       -    16       0    0      0      0     0
hpcs03.localhost   ok       -    16       0    0      0      0     0
hpcs04.localhost   ok       -    16       0    0      0      0     0
```

・STATUS：ノードの状態を示します。
　ok ・・・ 正常
　closed ・・・ ジョブスロットの上限まで使用している等の理由で、これ以上ジョブを受け付けない
　unavail ・・・ スケジューラーが正常に動作していない
　OS 起動直後やノードが起動していない場合にこのステータスになる場合があります
・JL/U：ユーザーごとにノードが処理できるジョブスロットの最大値
・MAX：ノードが処理できるジョブスロットの最大値
　デフォルトでは各ノードの CPU コア数と同じ値をセットします
・NJOBS：ノード上で開始されたジョブ（実行ジョブ、中断ジョブ、またはチャンクジョブを含む）によって使用されるジョブスロットの数
・RUN：ノード上の実行ジョブによって使用されるジョブスロットの数
・SSUSP：ノード上のシステム中断ジョブによって使用されるジョブスロットの数
・USUSP：ノード上のユーザー中断ジョブによって使用されるジョブスロットの数
　ジョブはユーザーまたは LSF 管理者によって中断されることがあります
・RSV：ノード上にジョブスロットを予約した保留ジョブによって使用されるジョブスロットの数

※ジョブスロットとは LSF・Lava スケジューラー上で各ノードが処理するジョブの単位です。
通常は各ノードの CPU コア数と同じ数をセットしています。

(3) bsub

バッチ実行用のジョブを投入して、それに一意の数値ジョブ ID を割り当てます。ジョブ、ノード、キュー、およびクラスタに関するすべての条件が満足されたとき、ジョブのすべての要件を満足するノード上でジョブを実行します。LSF･Lava がすべてのジョブをすぐに実行できない場合には、Lava スケジューリングポリシーによって実行順序が決定されます。また、ジョブは現在のシステム負荷に基づいて実行されます。

```
$ bsub␣-o␣~/logfile␣./a.out
Job <101> is submitted to default queue <normal>.
```

主なオプション

-o out_file：指定したファイルにジョブの標準出力を書き出します。
このファイルのパスを指定します。

-e err_file：指定したファイルにジョブの標準エラー出力を書き出します。
このファイルのパスを指定します。

-q "queue_name"：指定したキューの 1 つにジョブを投入します。
デフォルトでは normal キューを指定します。
ローカルクラスタで使用可能なキューのリストについては、bqueues を使用してください。

-m "host_name"：指定したノードの 1 つでジョブを実行します。複数のノードが候補である場合には、最も負荷の少ないノードでジョブを実行します。

-n：使用するジョブスロット数を指定します。ジョブが使用する CPU コア数と同じ値を指定してください。デフォルトでは 1 がセットされます。

(4) bjobs

ジョブの情報を表示します。デフォルトでは、自分の保留ジョブ、実行ジョブ、および中断ジョブの情報を表示します。

```
$ bjobs␣-u␣all
JOBID USER STAT QUEUE  FROM_HOST  EXEC_HOST    JOB_NAME     SUBMIT_TIME
202   hpc  RUN  normal  hpcs01.loc  hpcs04.loc   ./a.out      Jun 14 10:18
203   hpc  RUN  normal  hpcs01.loc  hpcs03.loc   ./test.sh    Jun 15 12:22
204   hpc  RUN  normal  hpcs01.loc  hpcs03.loc   ./job.sh     Jun 16 15:08
```

主なオプション

-u user_name ：指定したユーザーによって投入したジョブを表示します。

-u all ：すべてのユーザーが投入したジョブを表示します。

(5) bkill

指定したジョブを中止します。SIGINT と SIGTERM を送信してジョブに終了前の後処理を行い、SIGKILL を送信してジョブを中止します。

各シグナルを送信する間隔は設定ファイル configdir/lsb.params の JOB_TERMINATE_INTERVAL パラメータによって定義されます。(man␣lsb.params を参照)

```
$ bkill␣204
Job <204> is being terminated
```

主なオプション

-u user_name ： 指定したユーザーが投入しジョブをすべて中止します。

-u all ： すべてのユーザーのジョブを停止します。

(6) bqueues

キューの情報を表示します。デフォルトでは、すべてのキュー、キューの優先順位、キューの状態、ジョブスロット統計、およびジョブ状態統計が表示します。

```
$ bqueues
QUEUE_NAME  PRIO       STATUS  MAX JL/U JL/P JL/H NJOBS PEND RUN SUSP
priority    43    Open;Active    -    -    -    -     0    0    0    0
short       35    Open;Active    -    -    -    -     0    0    0    0
normal      30    Open;Active    -    -    -    -     0    0    0    0
```

・QUEUE_NAME：キューの名前
・PRIO：キューの優先度
・STATUS：キューの状態
・MAX：実行可能なジョブ数の制限
・JL/U： 実行できるジョブ数の制限
・JL/P：使用できるジョブスロット数の制限
・JL/H：使用できるノード台数の制限
・NJOBS：キューに割り当てられているジョブ数
・PEND：実行待ちのジョブ数
・RUN：実行中のジョブ数
・SUSP：中断されているジョブ数

主なオプション
-l ： キュー情報を長い複数行フォーマットで表示します。-l オプションを指定すると、次の追加情報が表示されます。
キューの説明、キューの特性と統計、スケジューリングパラメータ、リソース制限、スケジューリング ポリシー、ユーザー、ノード、ユーザー シェア、ウィンドウ、関連するコマンド、およびジョブ制御
-u user_name ： 指定したユーザーが使用できるキューを表示します

出荷時の設定では normal, short, priority の 3 つのキューが設定されています。
・デフォルトでは normal キューが使用されます。
・short キューは normal キューより優先度が高いが 1 時間でジョブが強制終了します。
・priority キューは最も優先度が高いキューです。

(7) bhist

ジョブに関する履歴情報を表示します。デフォルトでは、自分の保留ジョブ、実行ジョブ、および中断ジョブに関する情報を表示します。デフォルトでは情報をジョブ別にまとめます。デフォルトではイベントログファイル"INSTALL_DIR"/work/"Cluster_Name"/logdir/lsb.events を検索します。(man␣ lsb.events を参照) デフォルトでは 1 週間以内に発生したイベントを表示しますが、環境変数 LSB_BHIST_HOURS を設定すれば、これを代替の時間数に変更できます。

```
$ bhist␣-l␣508
Job <508>, user <hpc> ,Project <default>, Command <openmpi_wrapper -np 8
./a.out>
Wed JUN 16 18:58:54: Submitted from host <hpcs01.localhost>, to Queue
                 <normal>, CWD <$HOME /jobdir>, Output File <.>,
                 8 Processors Requested;
Wed JUN 16 18:59:02: Dispatched to 8/HostsProcessors <hpcs04.localhost>
                 <hpcs04.localhost> <hpcs04.localhost> <hpcs04.localhost>
                 <hpcs04.localhost> <hpcs04.localhost> <hpcs04.localhost>
                 <hpcs04.localhost>
Wed JUN 16 18:59:02: Starting (Pid 12269);
Wed JUN 16 18:59:05: Running with execution home </home/hpc>, Execution
                  CWD </home/hpc/jobdir>, Execution Pid <12269>;
```

主なオプション

-a：完成ジョブと未完成ジョブの情報を表示します。

-l：ロングフォーマット。追加の情報を表示します。-s と一緒に使用された場合には、各ジョブが中断された理由を表示します。

-s：中断ジョブの情報だけを表示します。

(8) bstop

実行中のジョブを中断します。順次ジョブには SIGCONT シグナルを送信し、並列ジョブには SIGTSTP シグナルを送信して、これらのジョブを中断します。

```
$ bstop␣305
Job <305> is being stopped
```

bstop で中断しているジョブはメモリを確保したままになるのでご注意下さい。

(9) bresume

bstop で止めたジョブを再開します。

```
$ bresume␣305
Job <305> is being resumed
```

(10) bpeek

指定した実行中のジョブの標準出力と標準エラー出力を表示します。デフォルトではコマンド cat を使用して出力を表示します。

このコマンドはジョブの進捗の確認等で使用できます。エラーが観測された場合には、ジョブを終了させることによって貴重なユーザー時間とシステムリソースを節約できます。

```
$ bpeek␣305
<< output from stdout >>
```

5　ジョブの投入

5.1　基本的なジョブの投入方法

```
$ bsub␣sleep␣100
```

UNIX コマンド sleep とその引数 100 をバッチジョブとして投入します。

```
$ bsub␣-q␣short␣-o␣my_output_file␣"pwd␣;␣ls"
```

UNIX コマンド pwd と ls をバッチ ジョブとして short というキューに投入し、ジョブ出力を my_output ファイルに格納します。

```
$ bsub␣-m␣"host1␣host3␣host8␣host9"␣my_program
```

my_program を投入して、候補ノード(host1,host3,host8 および host9)の 1 つで実行します。なお、-m オプションではノードのグループを指定することも可能です。

```
$ bsub␣-q␣"queue1␣queue2␣queue3"␣-c␣5␣my_program
```

候補キュー(queue1,queue2 および queue3)の 1 つに my_program を投入します。候補キューは、-c 5 によって指定される CPU 時間制限に基づいて選択されます。

```
$ bsub␣-I␣ls
```

ユーザーの端末に ls の出力を表示するバッチ対話型ジョブを投入します。

```
$ bsub␣-Ip␣vi␣myfile
```

myfile を vi で編集するためのバッチ対話型ジョブを投入します。

```
$ bsub␣-b␣20:00␣-J␣my_job_name␣./my_program
```

午後 8 時以降に実行する my_program を投入し、それにジョブ名 my_job_name を割り当てます。

5.2 並列ジョブの投入方法

並列ジョブを実行する場合は -n のオプションで使用する CPU コア数（ジョブスロット数）を指定します。

```
$ bsub␣-n␣8␣./my_program
```

8CPU コアを使用するジョブを実行します。

並列ジョブをクラスタ内へどのように分散するかを指定することができます。デフォルトで、Lava はジョブに必要な CPU コアを使用可能なノードで任意に指定します。bsub コマンドで-R オプションを使用するとリソース要件を指定できます。

span[hosts=1]：ジョブに割り当てられる全ての CPU コアは同じノードにすることを示します。

span[ptile=n]：ジョブに割り当てることができるノードあたりの最大 CPU コア数を示します。

```
$ bsub␣-n␣8␣-R␣"span[ptile=4]"␣./my_program
```

2 つのノードでそれぞれ 4CPU コアずつ確保してジョブを実行します。

```
$ bsub␣-n␣8␣-R␣"span[ptile=6]"␣./my_program
```

1 つのノードで 6CPU コアを確保し、2 つ目のノードでは 2CPU コアを確保してジョブを実行します。

```
$ bsub␣-n␣8␣-R␣"span[ptile=1]"␣./my_program
```

1 つのノードあたりに 1CPU コアずつを確保してジョブを実行します。

```
$ bsub␣-n␣8␣-R␣"span[hosts=1]"␣./my_program
```

１つのノードで 8CPU コアを確保してジョブを実行します。

Lava で並列ジョブを実行する際は指定した CPU コア数の空きが必要になります。例えば 8CPU コアを指定した場合に空きが 7CPU コアの場合であれば、他のジョブが終了するまで 8CPU コアを指定したジョブは実行されません

5.3 OpenMPI ジョブの投入方法

```
$ bsub␣-n␣16␣"openmpi_wrapper␣$cwd/my_program"
```

OpenMPI 実行用スクリプト openmpi_wrapper から実行します。openmpi_wrapper が環境変数 LSB_NCPU_HOSTS より machinefile を作成し、-np のプロセス数の引数を自動で指定して実行します。

上記例ではカレントディレクトリ $cwd にある my_program を 16 並列の MPI ジョブとして投入します。実行ファイルはパスを記述する必要があります。例えば $cwd/my_program , ./my_program , /usr/local/myprogram のようにご指定下さい。

5.4 プロセッサ予約

ジョブが頻繁に投入されるクラスタでは、CPU コア数を多く指定したジョブが CPU コア数を少なく指定したジョブに「先を越されて」CPU コアを占有されるために、必要とする CPU コア数を確保できず、なかなかジョブがスタートしない現象が発生します。

その現象の解決のために CPU コア数を多く指定したジョブが、一定時間 CPU コアを予約して他に使わせない仕組みがあります。この機能を「プロセッサ予約」といいます。デフォルトでは 150 秒間で必要な CPU コア数を確保します。

プロセッサを確保する時間を変更したい場合は、"INSTALL_DIR"/conf/lsbatch/lava/configdir/lsb.queues ファイルの SLOT_RESERV_TIME[n] を変更します。CPU コアは lsb.params ファイルで指定した MBD_SLEEP_TIME の n 倍秒間予約されます。(lsb.params の MBD_SLEEP_TIME はジョブをディスパッチする間隔の指定を行うパラメータです。)

```
Begin␣Queue
QUEUE_NAME␣=␣normal
PRIORITY␣=␣30
SLOT_RESERVE␣=␣MAX_RESERVE_TIME[15]
        :
        :
End␣Queue
```

5.5 GPU リソースの管理

IBM Spectrum LSF は設定を追加することで GPU リソースのスケジューリングに対応します。GPU リソースを指定した使用例について概説します。

(1) lsload から GPU 関連の情報を抜粋して出力します。

```
[hpc@hpcs01 ~]# lsload␣-I␣ngpus:ngpus_excl_t
HOST_NAME      status  ngpus ngpus_excl_t
node01             ok    2.0          1.0
node02             ok    2.0          1.0
node03             ok    5.0          4.0
node04             ok    5.0          4.0
```

ngpus：計算機に搭載している全 GPU 数
ngpus_excl_t：計算機で搭載している GPU の中で計算に使用できるものの数

(2) GPU を使用するジョブを LSF から投入する場合は次のように行います。

・1GPU を使用するジョブの投入

```
[hpc@hpcs01 ~]# bsub␣-R␣"rusage[ngpus_excl_t=1]"␣./gpujob
```

・2GPU を使用するジョブの投入

```
[hpc@hpcs01 ~]# bsub␣-R␣"rusage[ngpus_excl_t=2]"␣./gpujob
```

(3) 投入したジョブが使用する GPU リソースは bhist コマンドから確認ができます。

```
[hpc@hpcs01 ~]# bhist␣-al␣330
Job <330>, User <hpc>, Project <default>, Command <./nbody -benchmark -numdevic
                  es=1 -numbodies=1000000>
Wed Dec 17 17:49:49: Submitted from host <jc1>, to Queue <normal>, CWD <$HOME/g
                  pujob>, Output File <.>, Requested Resources <rusage[ngpus
                  _excl_t=1]>;
Wed Dec 17 17:54:10: Dispatched to <jc1>, Effective RES_REQ <select[type == loc
                  al] order[r15s:pg] rusage[ngpus_excl_t=1.00] >;
```

6　LSF・Lava デーモン

下記の 5 個のデーモンで動作します。

・LIM：ノード毎に実行されて負荷を監視する。
・RES：ノード毎に実行され、遠隔実行の要求を受け入れます。
・SBD：ノード毎に実行され、MBD からジョブの実行要求を受け取り、RES を使用して
　　ジョブを実行します。
・MBD：１つのクラスタに１つ実行されます。ユーザーからのジョブ実行要求を受け取り、
　　スケジューリングポリシーにしたがって、ジョブを投入していきます。
・PIM：ノード毎に実行され、ノードで実行されているジョブとそのプロセスを監視します。

※デーモンの起動の順番等は、デフォルトで設定されているので特に設定し直す必要は
　ありません。

7　設定ファイル

"INSTALL_DIR"/conf/configdir/"Cluster_Name"/lsbatch 以下に Lava の設定ファイルがあります。各ファイルの内容を概説します。

7.1　lsb.hosts

lsb.hosts ファイルには、クラスタ内のヘッドノードについてのノード関連の構成情報が格納されます。このファイルはオプションで、セクションもすべてオプションです。

HostGroup の項目でノードグループを定義します。ノードグループを定義しておけば、コマンド行はもちろん、他のノードグループ、ノードパーティションおよびキューの定義で使用できます。

先頭行は、GROUP_NAME と GROUP_MEMBER の必須キーワードから設定されます。それに続く行でグループ名を指定し、グループのメンバをリストします。

例

```
Begin␣HostGroup
GROUP_NAME␣GROUP_MEMBER
groupA␣(␣node01␣node02␣)
groupB␣(␣node11␣node12␣groupA␣)
End␣HostGroup
```

この例では 2 つのノードグループが定義されます。グループを設定することでジョブを投入するグループの指定をすることができます。

・groupA には node01 と node02 が含まれます。
・groupB に は groupA のすべてのノードに加え、node11 と node12 が含まれます。

7.2 lsb.users

lsb.users ファイルは、ユーザーグループの設定、さらにユーザーとユーザーグループのジョブスロット制限の設定に使用されます。このファイルはオプションです。

UserGroup の項目でユーザーグループを定義します。先頭行は、GROUP_NAME と GROUP_MEMBER の必須キーワードから設定されます。USER_SHARES キーワードはオプションです。それに続く行でグループ名を指定して、グループのメンバをリストし、オプションでシェア割当てをリストします。各行には 1 つのキーワードごとに 1 つのエントリを指定しなければなりません。エントリにデフォルト値を指定する場合は、空の括弧( ) またはダッシュ -を使用します。

```
Begin␣UserGroup
GROUP_NAME␣GROUP_MEMBER
bachelor␣(user01␣user02␣user03␣user04)
guest␣(user11␣user12␣user13␣user14)
teacher␣(user21␣user22)
End␣UserGroup
```

7.3 lsb.queues

lsb.queues ファイルは、Lava クラスタ内のバッチキューを定義します。それぞれのキューを Queue の項目で定義します。キューは様々なパラメータを使用して定義ができます。以下はキューで指定できる定義の一部です。

CPULIMIT

正規化 CPU 時間の最大制限値とデフォルト制限値を指定します(デフォルト制限値は任意指定です) 。ここで指定した CPU 時間の制限値は、このキューで実行されるジョブのすべてのプロセスに適用されます。ジョブが動的にプロセスを起動する場合、これらのプロセスで使用される CPU 時間が、ジョブの存続期間に渡って累積されます。30 秒未満で終了するプロセスは無視できます。デフォルトでは、デフォルト CPU 制限値が指定されると、ジョブレベルでの CPU 制限値なしで投入されたジョブは、デフォルト CPU 制限値に達すると強制終了されます。

HOSTS

このキューからのジョブを実行できるノード、ノードグループ、およびノードパーティションのスペースで区切られたリスト。いずれかのアイテムにプラス記号 (+) と正の数字を付けて、そのノード、ノードグループ、またはノードパーティションにジョブをディスパッチする優先順位を指定することができます。数字が大きいほど、優先順位が高くなります。ノードの優先順位が指定されていない場合は、優先順位は 0 であると想定されます。優先順位が同じレベルのノードは、負荷によって順序が決定されます。明示的にリストされていないすべてのノードを指定するには、キーワード others を使用します。キーワード all は明示的に除外されていないすべてのノードを指定するために使用します。

PRIORITY

キューの優先順位。値が大きいほど、他のキューと比較した場合の Lava ディスパッチ優先順位が高くなります。

USERS

キューにジョブを投入可能なユーザーやユーザーグループのリストを指定します。すべてのユーザーを指定するには、予約語 all 使用します。Lava クラスタ管理者はこのパラメータで指定されていなくても、このキューにジョブを投入し、任意のユーザーのジョブをこのキューに切り替えることができます。

例

```
Begin Queue
QUEUE_NAME    = bachelor
PRIORITY      = 30
NICE          = 20
CPULIMIT      = 180/node10  #node10 に 3 時間まで、それ以外は無制限
PROCLIMIT     = 8           #CPU は 8 個まで
USERS         = bachelor    #bachelor 所属のユーザーだけ使用できる
DESCRIPTION   = limited for bachelor.
End Queue

Begin Queue
QUEUE_NAME    = guest
PRIORITY      = 10          #ジョブの優先順位が低い
NICE          = 20
CPULIMIT      = 180         #CPU 時間制限 3 時間
PROCLIMIT     = 4           #CPU は 4 個まで
USERS         = guest       #guest 所属のユーザーだけ使用できる
HOSTS         = testnode    #testnode の中のマシンのみ選択
DESCRIPTION   = limited for guest.
End Queue

Begin Queue
QUEUE_NAME    = teacher
PRIORITY      = 45          #ジョブの優先順位が高い
NICE          = 20
CPULIMIT      = 1800        #CPU 時間制限 3 時間
PROCLIMIT     = 96          #96CPU まで
USERS         = teacher     #teacher 所属のユーザーだけ使用できる
HOSTS         = all         #全てのマシンを使用できる
DESCRIPTION   = all resources are reserved
End Queue
```

7.4 lsb.params

lsb.params ファイルは、スケジューラーによって使用される汎用パラメータを定義します。このファイルには、Parameters という名前のセクションが 1 つあるだけです。

マスター バッチデーモン(MBD) の初期設定に lsb.params ファイルを使用しますが、このファイルはオプションです。このファイルが存在しない場合は、Lava で定義されるデフォルトの設定が想定されます。

lsb.params ファイルで定義できるパラメータの中には、システム内のタイミングを制御するものがあります。デフォルトの設定では、バッチデーモンに加わる処理オーバーヘッドを最小限に抑えながら、長時間実行バッチ ジョブに対して十分なスループットも提供します。

DEFAULT_QUEUE

デフォルトで使用するキューのリスト (候補はあらかじめ lsb.queues で定義されていなければなりません)。

キューを明示的に指定せずにジョブを投入した場合、このリストの中で要求されたノードやキューの状態といった他の制約に従い、ジョブの仕様を最初に満足するキューにジョブを登録します。環境変数 LSB_DEFAULTQUEUE が設定されていればそのキューが優先されます。

MBD_SLEEP_TIME

ジョブのディスパッチ間隔。Lava が保留中のジョブをディスパッチする間隔。

MAX_SBD_FAIL

非応答のスレーブ バッチデーモン (SBD) に到達するための最大試行回数。

再試行が行われる間隔は、MBD_SLEEP_TIME で定義されます。MAX_SBD_FAIL で定義されている回数試行を繰り返しても MBD がノードに到達できない場合、このノードは使用不能状態であるとみなされます。

ノードが使用不要状態になると、MBD はこのノードで実行中のすべてのジョブが終了したものとみなし、再実行可能なすべてのジョブ (bsub -r オプションを使用して投入されたジョブ) は別のノードで再実行するようにスケジュールされます。

7.5 スケジューラーの更新

設定ファイル "INSTALL_DIR"/conf/configdir/"Cluster_Name"/lsbatch 以下を変更された場合はスケジューラーの更新を行って下さい。root で以下のように lsadmin と badmin コマンドを実行して正常終了することをご確認下さい。

```
# source␣/usr/share/lava/conf/lava.csh
# lsadmin␣reconfig
Checking configuration files ...
No errors found.

Do you really want to restart LIMs on all hosts? [y/n] y
Restart LIM on <hpcs01.hpc.co.jp> ...... done
Restart LIM on <hpcs02.hpc.co.jp> ...... done
Restart LIM on <hpcs03.hpc.co.jp> ...... done
Restart LIM on <hpcs04.hpc.co.jp> ...... done

# badmin␣reconfig
Checking configuration files ...
No errors found.
Reconfiguration initiated

Check configuration files …

No errors found.
Reconfiguration initiated
#
```

※ LSF の場合は最初に source␣/usr/share/lsf/conf/cshrc.lsf を実行して下さい。

付録A

A.1 HPC システムズ お問い合わせ先


弊社ホームページ　http://www.hpc.co.jp/support_index.html

サポート案内やお問い合わせの多い内容など様々な情報を掲載しております。
是非ご活用ください。

HPC システムズ株式会社
〒108-0022　東京都港区海岸 3-9-15 LOOP-X 8 階

HPC 事業部


【営業】03-5446-5531　【サポート】03-5446-5532

お電話によるサポート受付は祝日、弊社指定休日を除く月曜日から金曜日の 9:30～17:30 とさせて頂きます。


【FAX】03-5446-5550


【電子メール】hpcs_support@hpc.co.jp