LSF クイックスタート

rev.3.1

1　基本的なコマンド

(1)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、ホスト、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ホスト上でジョブを実行します。LSFがすべてのジョブをすぐに実行できない場合には、LSF スケジューリング ポリシーによってディスパッチの順序が決定されます。ジョブは、現在のシステム負荷に基づいて起動および中断されます。

主なオプション

-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。

-o *out_file*

ファイル パスを指定します。ジョブの標準出力を指定のファイルに追加します。そのファイルが存在しない場合、またはシステム障害によってファイルに書き込みできない場合には、出力をメールで送信します。

-q *"queue_name ..."*

指定したキュー の 1 つにジョブを投入します。キュー が 1 つしかない場合には、引用符を省略してもかまいません。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*[+[*pref_level*]] | *host_group*[+[*pref_level*]] ..."

指定したホスト の 1 つでジョブを実行します。デフォルトでは、複数のホストが候補である場合には、最も負荷の少ないホストでジョブを実行します。

(2)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。

主なオプション

-u *user_name* | **-u *user_group*** | **-u all**

指定したユーザによって投入されたジョブだけを表示します。キーワード allはすべてのユーザを指定します。

(3)bkill

デフォルトでは、指定したジョブを中止するための一連のシグナルを送信します。UNIX では、SIGINT と SIGTERM を送信してジョブに終了前の後処理を行うチャンスを与えてから、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時間間隔は、lsb.params(5) の JOB_TERMINATE_INTERVALパラメータによって定義されます。

主なオプション

-u *user_name* | **-u *user_group*** | **-u all**

指定したユーザまたはユーザ グループ (bugroup(1) を参照) によって投入されたジョブ (あるいは、予約ユーザ名である all を指定した場合には、すべてのユーザによって投入されたジョブ) にだけ作用します。

(4)bhosts

ホストの情報を表示します。デフォルトでは、全ホストの情報 (ホスト名、ホスト状態、ジョブ スロット制限、およびジョブ状態統計) を返します。

(5)bqueues

キューの情報を表示します。デフォルトでは、すべてのキューについて次の情報を返します。キューの名前、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。

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、ホスト、ユーザ 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* | -u *user_group* | -u all

指定したユーザやユーザ グループからジョブを受け付けることのできるキューを表示します (ユーザ グループのリストについては、bugroup(1) 参照) 。キーワード ™ all™ が指定された場合には、全ユーザからジョブを受け付けることのできるキューを表示します。

(6)bpeek

このコマンドが呼び出されるまでに、未完成ジョブ の 1 つによって生成された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、ジョブの進捗を監視してエラーを識別するために役立ちます。エラーが観測された場合には、エラーのあるジョブを終了させることによって、貴重なユーザ時間とシステム リソースを節約できます。

(7)lsload

ホストの負荷情報を表示します。負荷情報は、ホストごとに、またはリソースごとに表示できます。
デフォルトでは、ホストごとに、ローカル クラスタ内の全ホストの負荷情報を表示します。
デフォルトでは、生（ 未処理） の負荷インデックスを表示します。
デフォルトでは、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。

(8)bhist

LSF 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、CPU 時間は正規化されません。デフォルトでは、LSF システムによって使用されているイベント ログ ファイルである$LSB_SHAREDIR/*cluster_nam*e/logdir/lsb.events を検索します (lsb.events(5) を参照)。デフォルトでは、ここ 1 週間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。

-a

完成ジョブと未完成ジョブの情報を表示します。このオプションは、-d-p-sおよ び -r を無効にします。

-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。

２.１　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、候補ホス ト (host1 、host3 、host8 、および host9)の 1 つで実行します。
% **bsub -q "queue1 queue2 queue3" -c 5 my_program**
　　候補キュー (queue1queue2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 を編集するためのバッチ対話型ジョブを投入します。
% **bsub -Is csh**
　　対話型シェルとして csh を起動する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。

２.２　並列ジョブの投入方法

% **bsub –n 4 pam –g 1 $LSF_BINDIR/mpichp4_wrapper $cwd/my_program**

カレントにある my_program を 4 並列 MPI ジョブとして投入します。Pam 以降は絶対パスで指定する必要があります。Pam を利用すると、MPI ジョブの生成したプロセスに異常が発生した場合に、その MPI ジョブの生成したプロセスを LSF デーモンがすべてクリーンにします。MPI ジョブの生成したプロセスに異常が発生した場合、親プロセスがなくなることによって、その子プロセスがゾンビプロセスになることがあります。MPI ジョブの生成したプロセスを LSF デーモンがすべてクリーンすることで、ゾンビプロセスが CPU やメモリを消費することを未然に防ぐことが出来ます。動的な並列実行ノードの指定も行います。MPICH が正常に動作する環境が前提条件になります。MPICH については MPICH クイックスタートを参考にしてください。

LSF にて MPI ジョブを実行する際には指定した CPU 数の空きが必要になります。例えば 4cpu 指定した場合に、空きが 3cpu の場合ですと他のジョブが終了するまで 4cpu を指定したジョブは実行されません。

注意

・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 数を多く指定したジョブが、CPU 数を少なく指定したジョブに「先を越されて」CPU を占有されるために、必要とする CPU 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。その現象の解決には、CPU 数を多く指定したジョブが、一定時間 CPU を予約して、他に使わせない仕組みが必要です。この機能を「プロセッサ予約」といいます。デフォルトでは、5 分間で必要な CPU 数を確保できなかった場合には、キューイングされたジョブが強制修了されます。プロセッサを確保する時間を変更したい場合は、

/usr/share/lsf/conf/lsbatch/<cluster>/configdir/lsb.queues の SLOT_RESERV_TIME[n]エントリを変更します。CPU は、MBD_SLEEP_TIME*n 秒間、予約されます。MBD_SLEEP_TIME は、lsb.params に定義されていて、ジョブをディスパッチする間隔を表します。

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

・並列ジョブの局所性の問題

並列ジョブのために選択されたCPUをクラスタ内のホストにどのように分散するかを指定しなければいけない場合があります。デフォルトで、LSFはジョブに必要なCPU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次のリソース要件を指定できます。

span[hosts=1]　このジョブに割り当てられるCPUは同じホストになければいけないことを示します。

span[ptile=n]　ホストで処理されるCPU数とは関係なく、このジョブに割り当てられる各ホストの最大CPU数を示します。

例　# bsub - n 4 - R “span[ptile=2]” “mpijob mpirun a.out”

　　2つのホストでそれぞれ2CPUずつ使用してジョブを実行します。

例　# bsub - n 4 - R “span[ptile=3]” “mpijob mpirun a.out”

　　2つのホストで最初のホストでは、3つのCPUを使用し、2つ目のホストでは1つのCPUを使用します。

例　# bsub - n 4 - R “span[ptile=1]” “mpijob mpirun a.out”

　　複数のプロセッサが使用可能なホストが含まれていたとしても、各ホストで1CPUずつしか使用されません。

例　# bsub - n 4 - R “span[hosts=1]” “mpijob mpirun a.out”

　　4つのプロセスのためにCPUが4つ以上あるホストでジョブが実行されます。

・ホストの指定

クラスタを構成するホストに性能差(CPU、メモリetc)がある場合にはMPIジョブを効率良く行なうにはスペックが同一のホストで行なわないと、スペックの劣るホストを混ぜて実行した場合にはスペックの劣るホストが足を引っ張ります。その場合にはジョブ投入時に同一スペックのホストで実行するようにホストの指定を行ないます。

例　# bsub - n 4 - m ‘hpc1 hpc2 hpc3 hpc4’ “mpijob mpirun a.out”

　　hpc1,hpc2,hpc3,hpc4のホストでジョブを実行します。

またLSFクイックスタートにてLSFホストのグループ分けを行なっている場合にはグループ指定することもできます。

例　# bsub - n 4 - m group1 “mpijob mpirun a.out”

　　group1のホストから4CPUを確保してジョブを実行します。

・mpijob スクリプトの利用

mpijobスクリプトを用いると動的な並列実行ノードの指定のみを行ないます。mpichが正常に動作する環境が前提条件になります。またジョブの投入はbsubコマンドを併用して行ないます。bsubコマンドについてはLSFクイックスタートを参考にしてください。

bsub - n [cpu数] “mpijob mpirun [実行プログラム]”

実行例

CPUを4つ使用し、outファイルにbsubコマンドの出力を指定し、a.outというプログラムを実行する場合

bsub - o out - n 4 “mpijob mpirun a.out”

３　設定ファイル

※下記の設定を LSF に反映させるのは、各種デーモンの再起動が必要です。具体的な方法は「５　LSF で動作するデーモンについて」を御覧ください。

(1)lsb.queues

lsb.queues ファイルは、LSF クラスタ内のバッチ キューを定義します。このファイルはオプションです。キューが設定されていない場合は、すべてのパラメータをデフォルト値に設定して、LSF が default という名前のキューを作成します。

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ホスト、ホスト グループ、およびホスト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、そのホスト、ホスト グループ、またはホスト 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。ホスト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ホストは、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ホストを指定するには、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ホスト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LSF ディスパッチ優先順位が高くなりす。

USERS

キューにジョブを投入可能なユーザやユーザ グループのリストを指定しますすべてのLSFユーザを指定するには、予約 all を使用します。LSF 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たり (FAIRSHARE パラメータが未定義の場合のみ) 、任意のユーザのジョブをこのキューに切り替えることができます。

FAIRSHARE

キューレベルのフェアシェアを有効化し、シェアの割当てを指定します。シェアを割り当てられたユーザだけが、このキューにジョブを投入できます。

構文

FAIRSHARE = USER_SHARES[*[user, number_shares*] ...]

- **少なくとも** 1 **つのユーザ シェアの割当てを指定します。**
- **上記に示すように、各リストは角括弧で囲みます。**
- **上記に示すように、各ユーザ シェアの割当ては角括弧で囲みます。**
- *user*

キューを使用できるようにも設定されているユーザを指定します。次のようなシェアの割当てを行うことができます。

- **一人のユーザに割り当てる** *(user_name* **を指定する)。**
- **グループ内のユーザごとに割り当てる** (*group_name@* **を指定する) か、またはグループ全体に割り当てる** *(group_name* **を指定する)**
- **他のシェアの割当てに含まれていないユーザごとに割り当て る** (**キーワード** default **を指定する) か、またはこれらのユーザ全員を一グループとして割り当てる** (**キーワー ド** others **を指定する)**

デフォルトでは、グループ全体にリソースが割り当てられている場合、グループ メンバは FCFS **に従い、リソースを競合します。階層的フェアシェアを使用すれば、グループ メンバ間でシェアを細分化することができます。**

グループ メンバー人一人にリソースが割り当てられている場合は、シェアの割当ては再帰的です。そのグループとすべてのサブグループのメンバは、階層的フェアシェアのポリシーに関係なく、FCFS **スケジューリングに従って、常にリソースを競合します。**

(2)lsb.hosts

lsb.hosts ファイルには、クラスタ内のサーバ ホストについてのホスト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

機能説明

ホスト グループを定義します。ホスト グループを定義しておけば、コマンド行はもちろん、他のホスト グループ、ホストパーティション、およびキューの定義で使用できます。ホスト グループ名を指定することは、グループ内のすべてのホスト名をリストするのとまったく同じです。

構造

先頭行は、GROUP_NAME と GROUP_MEMBER の 2 つ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。ホスト グループの合計数は、MAX_GROUPS を超えることはできません 。

例 1

```
Begin HostGroup
GROUP_NAME GROUP_MEMBER
groupA (hostA hostD)
groupB (hostF groupA hostK)
End HostGroup
```

この例では、次 の 3 つのホスト グループが定義されます。

・・groupA に は hostA と hostD が含まれます。

・・groupB に は groupA のすべてのホストに加え、hostF と hostK が含まれます。

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。

% **bsub -m groupA my_program**

グループ A のいずれかのホストで my_program を実行します。

(3)lsb.users

lsb.users ファイルは、ユーザ グループの設定、ユーザとユーザ グループに対する階層的フェアシェアの設定、さらにユーザとユーザ 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

機能説明

ユーザ グループを定義します。ユーザ グループ名は、コマンド行はもちろん、他のユーザ グループやキューの定義も使用できます。

構造

先頭行は、GROUP_NAME と GROUP_MEMBER の 2 つ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。

例

```
Begin UserGroup
GROUP_NAME GROUP_MEMBER
groupA (user1 user2 user3 user4)
groupB (groupA user5)
groupC (!)
End UserGroup
Begin UserGroup
GROUP_NAME GROUP_MEMBER USER_SHARES
groupB (user1 user2) ()
groupC (user3 user4) ([User3,3] [User4,4])
groupA (GroupB GroupC User4) ([User4,1] [default,10])
        End UserGroup
```

(4)lsb.params

lsb.params ファイルは、LSF Batch 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。MBD が初期設定に 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、LSF 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、LSF Batch 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 キュー候補のスペースで区切られたリスト（ 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しないで LSF にジョブを投入した場合、環境変数LSB_DEFAULTQUEUE が設定されていないと、LSF はこのリストの中で、要求されたホスト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。

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

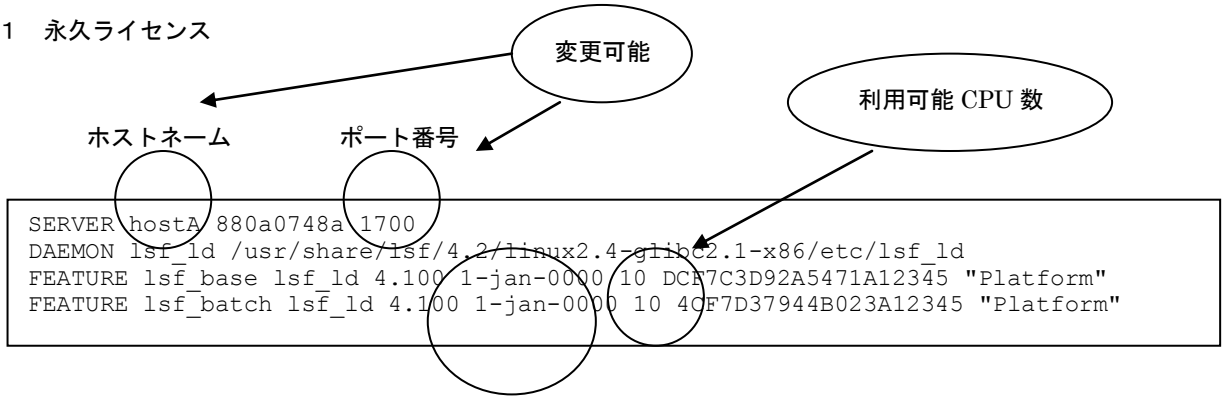
MAX_SBD_FAIL
非応答のスレーブ バッチ デーモン（SBD）に到達するための最大試行回数。再試行が行われる間隔は、MBD_SLEEP_TIME で定義されます。MAX_SBD_FAIL で定義されている回数試行を繰り返しても MBD がホストに到達できない場合、このホストは使用不能状態であるとみなされます。ホストが使用不要状態になると、MBD はこのホストで実行中のすべてのジョブが終了したものとみなし、再実行可能なすべてのジョブ（bsub -r オプションを使用して投入されたジョブ）は別のホストで再実行するようにスケジュールされます。

４　ライセンス

/usr/share/lsf/conf/license.datに置かれています。

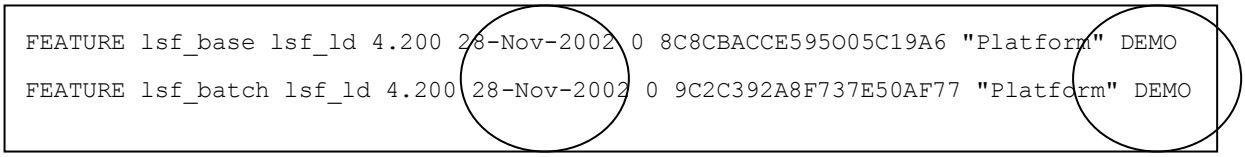
ライセンスファイルの構造

例１　永久ライセンス



ライセンス期限（期限のないことを示している。）

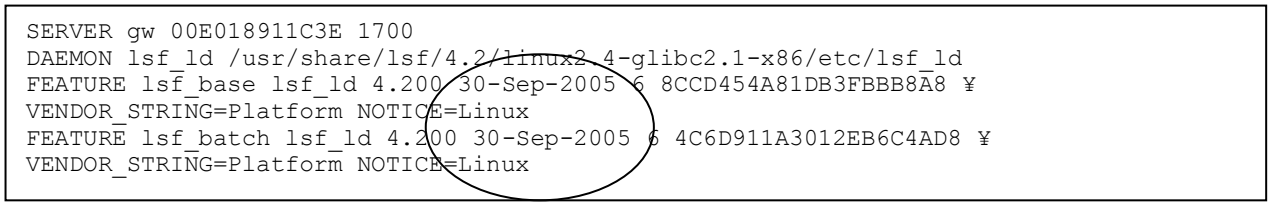
例２　デモライセンス



ライセンス期限（2002/11/28 まで）

デモライセンスである
ことを示している。

例３　期限付きライセンス



ライセンス期限（2005/09/30 まで）

ライセンス更新手順

注意点としてこの作業はライセンス更新時に行います。クラスタの再起動は行う必要はありません。また実行中、投入したジョブはそのまま実行されますが、ライセンス入れ替え中はLSF 関連コマンドを実行できないので新たにジョブを投入することができません。ライセンスファイルが届きましたら、ライセンスサーバーの/root の下にライセンスファイルをコピーしてください。

・届いたライセンスファイルの確認を行います。

ライセンスファイル例

SERVER p4l ******** 1700
DAEMON lsf_ld /usr/local/lsf/etc/lsf_ld
FEATURE lsf_base lsf_ld 4.200 30-Sep-2005 6 8CCD454A81DB3FBBB8A8 ￥
VENDOR_STRING=Platform NOTICE=Linux
FEATURE lsf_batch lsf_ld 4.200 30-Sep-2005 6 4C6D911A3012EB6C4AD8 ￥
VENDOR_STRING=Platform NOTICE=Linux

一行目の"p4l"と"1700"の間に12桁の英数字(ライセンスサーバのeth0のMacアドレス)
があります。
現在運用中のクラスタのライセンスファイルと比べて同じかどうかの確認を行ってく
ださい。
異なる場合には間違ったライセンスファイルが届いている可能性がありますので、お
手数ですが弊社宛にご連絡をお願いします。
日付の次に書いてある数字がライセンスにて有効なCPU数になります。
increamentにて増設している場合には分かれて記述されている場合もあります。
こちらも既存のライセンスファイルと比べてCPU数が同じかどうかの確認を行ってく
ださい。

・届いたライセンスファイルの2ヵ所(ホスト名、パス)を編集を行います。

１．ホスト名
ライセンスファイル例の一行目のSERVERの右のp4lを、
LSFライセンスサーバーのホスト名に書き換えてください。

２．パス
ライセンスファイル例の二行目のDAEMONのlsf_ldのパスを下記のように書き換え
てく
ださい。

Pentium4/Xeonの場合、
DAEMON lsf_ld /usr/share/lsf/6.1/linux2.6-glibc2.3-ia32e/etc/lsf_ld

Itanium2の場合、
DAEMON lsf_ld /usr/share/lsf/6.1/linux2.6-glibc2.3-ia64e/etc/lsf_ld

と書き換えてください。

・ライセンスの入れ替え
ファイル名の変更を行います。

```
# cd /root/work
# mv '*****.txt' license.dat
```

現在のライセンスファイルをリネームして残しておきます(作業を行った日時をファイル名に入れるのが良いです。)

```
# cd /usr/share/lsf/conf
# cp license.dat license.dat.20050401
```

※ 2005 年 4 月 1 日に作業した場合です

新しいライセンスファイルをもとのライセンスファイルへ上書きします。

```
# cp /root/work/license.dat /usr/share/lsf/conf/license.dat
```

次に、root ユーザで以下のコマンドを実行してください。

```
# /etc/init.d/lsf stop
# /etc/init.d/lsf_license stop
（↑のコマンドが無い場合は、/etc/rc.d/lsf_license stop）
# killall pim
# ps -awx | grep lsf
# kill -9 ↑で得たプロセス ID
# /etc/init.d/lsf_license start
（↑のコマンドが無い場合は、/etc/rc.d/lsf_license start）
# /etc/init.d/lsf start
# lmreread -c /usr/share/lsf/conf/license.dat
```

以上にて、ライセンス更新作業が終了となります。
bhosts　等のコマンドにて、動作確認をお願い致します。

５　LSF で動作するデーモンについて

※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。

LSF は下記の 5 コのデーモンとライセンスサーバーで動作します。
LIM　　ホスト毎に実行されて負荷を監視する。
RES　　ホスト毎に実行され、遠隔実行の要求を受け入れます。
SBD　　ホスト毎に実行され、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　　ホスト毎に実行され、ホストで実行されているジョブとそのプロセスを監視します。

(1)ライセンスサーバー

ライセンスサーバーを立ち上げてから、LSF の各種デーモンを起動させる必要があります。
①ライセンスサーバーの起動
```
# /etc/rc.d/lsf_license start
```
②各種デーモンの起動
```
# /etc/rc.d/lsf start
```

(2)ライセンスサーバー以外のホスト

ライセンスサーバーが起動した後、各種デーモンを起動させる必要があります。
```
# /etc/rc.d/lsf start
```

(3)設定ファイルを変更した場合に、その変更を LSF に反映させる方法。

各種デーモンの再起動を行います。次の２つのコマンドを実行します。
```
# lsadmin reconfig
```
出力例
```
# lsadmin reconfig

Checking configuration files ...
No errors found.

Do you really want to restart LIMs on all hosts? [y/n] y
Restart LIM on <gaugw.hpc.co.jp> ...... done
Restart LIM on <pen1.hpc.co.jp> ...... done
Restart LIM on <pen2.hpc.co.jp> ...... done
Restart LIM on <pen3.hpc.co.jp> ...... done
Restart LIM on <pen4.hpc.co.jp> ...... done
Restart LIM on <pen5.hpc.co.jp> ...... done
Restart LIM on <xeon1.hpc.co.jp> ...... done
Restart LIM on <xeon2.hpc.co.jp> ...... done
```
```
# badmin reconfig
```
出力例
```
# badmin reconfig

Checking configuration files ...

No errors found.

Reconfiguration initiated
```