ANCL

ShareTask 5.5

(doc#08)

ライセンスチェック設定マニュアル 1.2

アンクル
2014 年

改訂履歴

日付	版	内容
2014/4/19	1.0	初版
2014/4/20	1.1	6 章 ジョブ属性の参照方法を加筆
2014/4/21	1.2	7 章 デバッグを加筆
		表 6.0.2 にホスト名の参照方法を加筆

目次

1 はじめに 7

2 アプリケーションチェック 7

3 UserDefine.filter とは 7

4 UserDefine.filter の基礎 8
4.1 Perl 言語 8
4.2 UDF の処理の流れ 8

5 UDF の具体例 8

6 ジョブ属性の参照方法 14

7 デバッグ 14

表目次

6.0.1 ジョブ属性に対応する変数 14
6.0.2 問い合わせしているエージェントの情報 14

1 はじめに

本文書は、ShareTask のジョブ実行開始の可否判定において、アプリケーションライセンスの過不足検査を組み込むための手順を説明します。

2 アプリケーションチェック

アプリケーションベンダーから購入したアプリケーションが実行を開始できるためには、計算ノード、CPU、メモリといったハードウェア資源が充足されるだけでなく、アプリケーションベンダーが提供するアプリケーションライセンス（以降、ライセンス）も充足される必要があります。
したがって、バッチジョブスケジューリングにおいては、ハードウェア資源と、ライセンスの両方の充足を確認した上で、ジョブを実行開始する制御が必要になります。
ShareTask のジョブスケジューラーでは、製品の標準機能としては、ハードウェア資源の充足のみを確認します。したがって、この標準機能だけでは、ライセンスを必要とするアプリケーションのジョブについては、ライセンスが不足しているにもかかわらずに、ジョブを実行開始してしまう状況が起こりえます。
そこで、実行開始対象であるジョブについて、そのジョブが必要としているライセンスの過不足を検査する機能を追加できるようになっています。この検査機能の追加する部位を UserDefine.filter と呼びます。

3 UserDefine.filter とは

UserDefine.filter（以降、UDF）は、下記にある Perl スクリプトです。

```
/home/sharetask/sharetask_server/conf/plugin/UserDefine.filter
```

UDF は、ジョブ待ち行列（ジョブキュー）から実行開始対象のジョブを選択する一連の判定ロジックのひとつとして解釈実行されます。この UDF の戻り値 (return value) が 1 であれば、ジョブ実行開始の条件を満たしていることを意味しますので、後続の判定ロジックの評価が行われます。この UDF の戻り値 (return value) がゼロであれば、ジョブ実行開始の条件を満たしていないことを意味しますので、

後続の判定ロジックの評価は行われず、当該ジョブの実行開始は次回の判定へ見送られます。

4 UserDefine.filter の基礎

4.1 Perl 言語

UDF は、純粋な Perl スクリプトです。Perl 言語について基礎的な知識が必要になります。

4.2 UDF の処理の流れ

UDF の中の処理の流れは、以下の通りです。

1. 判定対象のジョブが必要とするライセンスの種類と数を決定する
2. ライセンスサーバーからライセンスの割り当て状況を取得する
3. ライセンスが充足されるか判定して、充足されれば 1 を戻り値とし、充足されなければゼロを戻り値として処理を終了する

5 UDF の具体例

具体的なコーディングを 10 ページから示します。
6 章とあわせて読むと理解しやすいと思います。

■8 行目 8 行目から始まる if-elsif-else ブロックでは、判定対象のジョブのアプリケーションの種類を取得しています。$prog という変数には、ジョブのプログラム名が与えられていますので、この例では、abaqus という文字列を含んでいたら ABAQUS のライセンスで検査を行います。同様に、ls あるいは mpp という文字列を含んでいたら、LS-DYNA のライセンスで検査を行います。

■17 行目 17 行目から始まる関数は、ABAQUS のライセンス検査を行う関数です。25 行目で、UDF_get_lic という関数を呼び出して現在のライセンスの割り当て状況を取得しています。その引数は、ABAQUS のライセンスフィーチャー名 abaqus です。この関数の戻り値、$used には、現在割り当て済み（使用中）のラ

イセンス数、$total には、ライセンスマネージャーに設定されているフィーチャー abaqus の全ライセンス数（使用中と未使用の合計）です。

■28 **行目**　28 行目で、判定対象のジョブの並列度$ncpu と、その並列度が要求するライセンス数（トークン数）$nreq を取得しています。

■31 **行目**　31 行目で、求めている$remains の値は、仮に検査対象のジョブを実行開始した場合に、未使用のライセンスがいくつ残るかです。

■33 **行目**　33 行目から始まる if ブロックで、$remains の値がゼロ以上であれば、検査対象のジョブを実行開始した場合に、ライセンスが充足されるので、return 1; とすることにより、実行可能であることを呼び出し元に返します。
一方、$remains が負の値であれば、ライセンスが充足できないことを意味しますから、return 0; とすることにより、実行してはいけないことを呼び出し元に返します。

■46 **行目**　46 行目から始まる関数は、アプリケーションが LS-DYNA の場合の検査を行うものです。関数内部の処理の流れは、ABAQUS の場合と同じですので、容易に理解できます。

■76 **行目**　76 行目、90 行目から始まる関数が、ライセンスマネージャーに問い合わせて、ライセンスの割り当て状況を取得します。

■91 **行目**　91 行目は、ライセンスマネージャーのロケーションを示すホストとポート番号を定義しています。このホストとポート番号を、ユーザーの実環境に合わせてください。

■92 **行目**　92 行目は、ライセンスマネージャーに問い合わせるコマンドを呼び出しています。この例では、Flex ライセンスマネージャーの場合ですので、もし異なるライセンスマネージャーをお使いの場合は、それに合わせて修正してください。また、101 行目付近で、コマンドの出力行をパターンマッチしているところも Flex の場合ですので、状況に応じて修正してください。コマンドをマニュアルで実行して、どのような出力が得られるのかを目で確かめるとよいでしょう。

■127 **行目**　127 行目、152 行目の関数は、次のような状況に対する解決策です。ジョブが実行開始されてから、実際にライセンスマネージャーからそのジョブへライセンスが割り当てられるまでに遅延があります。その間隙を突いて次のジョブの判定が行われてしまうと、実際にはライセンスが不足しているのに、充足できると誤判定してしまいます。この遅延の間は、ライセンス検査を必ず失敗させることにより、この誤判定を防ぎます。

■160 **行目**　最後の行

```
1;
```

は、とても重要です。これを忘れないようにしてください。

```
 1 #
 2 # ShareTask User Define Filter
 3 # Sample Script
 4 #
 5
 6 warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue);
 7
 8 if($prog =~ /abaqus/i){
 9   return UDF_check_abaqus($prog);
10 }elsif($prog =~ /^ls/i || $prog =~ /^mpp/){
11   return UDF_check_lsdyna($prog);
12 }else{
13   return 1;
14 }
15
16 #-----------------------------------------------------------------------------
17 sub UDF_check_abaqus {
18   my $prog = shift;
19   my $feature = "abaqus";
20   my $used;
21   my $total;
22
23   return 0 if(!UDF_check_guardtime($feature, 60)); # GUARDTIME MECHANISM
24
25   ($used, $total) = UDF_get_lic($feature);
26   warn("UDF: feature=" . $feature . " used=" . $used . " total=" . $total);
27
28   my $ncpu = $target_job->{required_ncpu};
29   my $nreq = int(5*exp(0.422*log($ncpu)));
30
31   my $remains = $total - $used - $nreq;
32
33   if($remains >= 0){
34     warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue .
35          " feature=" . $feature . " nreq=" . $nreq . " remain=" . $remains . " return 1");
36     UDF_update_guardtime($feature); # GUARDTIME MECHANISM
37     return 1;
38   }else{
39     warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue .
40          " feature=" . $feature . " nreq=" . $nreq . " remain=" . $remains . " return 0");
41     return 0;
42   }
43 }
44
45 #-----------------------------------------------------------------------------
46 sub UDF_check_lsdyna {
47   my $prog = shift;
48   my $feature = "LS-DYNA(total)";
49   my $used;
50   my $total;
51
52   return 0 if(!UDF_check_guardtime($feature, 30)); # GUARDTIME MECHANISM
53
54   $used = 0;
55   ($used, $total) = UDF_get_lic($feature);
56
57   warn("UDF: feature=" . $feature . " total used=" . $used . " total=" . $total);
58
59   my $nreq = $target_job->{required_ncpu};
60
61   my $remains = $total - $used - $nreq;
```

```
  62
  63   if($remains >= 0){
  64     warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue .
  65          " feature=" . $feature . " nreq=" . $nreq . " remain=" . $remains . " return 1");
  66     UDF_update_guardtime($feature); # GUARDTIME MECHANISM
  67     return 1;
  68   }else{
  69     warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue .
  70          " feature=" . $feature . " nreq=" . $nreq . " remain=" . $remains . " return 0");
  71     return 0;
  72   }
  73 }
  74
  75 #-------------------------------------------------------------------------
  76 sub UDF_get_lic {
  77   my $feature = shift;
  78   my @lines = split(/\n/, UDF_get_lic_list());
  79   foreach my $line (@lines){
  80     #warn($line);
  81     my @w = split(/[ \/]/, $line);
  82     if($w[0] eq $feature){
  83       warn("feature=" . $w[0] . " used=" . $w[1] . " total=" . $w[2]);
  84       return $w[1], $w[2];
  85     }
  86   }
  87 }
  88
  89 #-------------------------------------------------------------------------
  90 sub UDF_get_lic_list{
  91   my $LM_SERVER = "12700@mylmhost";       # your license manager location
  92   my $lines = `lmutil lmstat -c $LM_SERVER -a`;
  93
  94   @list = split(/\n/, $lines);
  95
  96   my $out = "";
  97
  98   foreach(@list){
  99     my $line = $_;
 100     my $feature = "undef";
 101     if($line =~ /^Users of/){
 102       # EXAMPLE of $line
 103       # Users of standard:  (Total of 10 licenses issued;  Total of 0 licenses in use)
 104       if($line =~ /(.*):(.*)/){
 105         my $header = $1;
 106         my $value  = $2;
 107         if($header =~/Users of (.*)/){
 108           $feature = $1;
 109         }
 110         if($value =~/(.*);(.*)/){
 111           my $first = $1;
 112           my $second = $2;
 113           if($first =~/Total of (.*) license[s] issued/){
 114             $total = $1;
 115           }
 116           if($second =~/Total of (.*) license[s] in use/){
 117             $inuse = $1;
 118           }
 119         }
 120       }
 121     }
 122     $out .= "$feature $inuse/$total\n"
```

```
123   }
124 }
125
126 #-------------------------------------------------------------------------
127 sub UDF_check_guardtime {
128   my $feature   = shift;
129   my $guardtime = shift;
130   my $timefile  = "/tmp/UDF_last_return1_" . $feature;
131   my $now       = time();
132   if( -e $timefile ){
133     my $mtime = (stat($timefile))[9];
134     if( $now - $mtime <= $guardtime ){
135       # has not elapsed for $guardtime after the last return 1.
136       warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue .
137            " " . ($now - $mtime) . " <= " . $guardtime . ", return 0");
138       return 0;
139     }else{
140       return 1;
141     }
142   }else{
143     # create $timefile
144     open my $fh, ">", $timefile;
145     close $fh;
146     warn("UDF: job=" . $job_id . " prog=" . $prog . " queue=" . $queue . " "
       . $timefile . " created.");
147     return 1;
148   }
149 }
150
151 #-------------------------------------------------------------------------
152 sub UDF_update_guardtime {
153   my $feature  = shift;
154   my $timefile = "/tmp/UDF_last_return1_" . $feature;
155   my $now      = time();
156   utime $now, $now, $timefile;
157 }
158
159 # do not remove '1;'
160 1;
```

6 ジョブ属性の参照方法

UDF のプログラミングにおいては、検査対象のジョブについて、そのプログラム名、並列度（CPU 数）などの属性値を参照する必要があります。UDF のコードが実行されるときには、表 6.0.1 に示すように各属性に対応する変数が定義されていますので、これらを参照することで多様な検査手順をプログラミングできます。

属性	変数
ジョブ ID	`$job_id`
プログラム名	`$prog`
キュー名	`$queue`
並列度	`$target_job->{required_ncpu}`

表 6.0.1　ジョブ属性に対応する変数

さらに、当該ジョブについて問い合わせをしているエージェントについても情報を取得することができます。（表 6.0.2）

属性	変数
エージェント ID	`$agent_id`
エージェント情報の構造体	`my $agent_attr = $self->agent_attr($agent_id);`
エージェントのホスト名	`$agent_attr->{AA_HOST_NAME}`
エージェントが割り当て可能な CPU 数	`$agent_attr->{AA_PROCS_FREE}`

表 6.0.2　問い合わせしているエージェントの情報

7 デバッグ

UDF の動作を確認するには、httpd(apache) のエラーログファイル
`/var/log/httpd/error_log`
を確認してください。

注意

エラーログファイルのファイルパスは、httpd のインストールによって異なる場合があります。

UDF のソースコードの各所に記述されている warn 関数の引数文字列は、このエラーログファイルに書き出されます。このメッセージから、UDF が期待通りの判定を行っているかを確認することができます。

以上