ワークステーションにおける大規模計算 Job の運用管理に関する一考察 (1)

武蔵 泰雄 、杉谷 賢一*

熊本大学総合情報処理センター
〒 860 熊本市黒髪 2-39-1
TEL 096-342-3915 , *096-342-3914 , FAX 096-342-3829
musashi@gpo.kumamoto-u.ac.jp, sugitani@eecs.kumamoto-u.ac.jp

概要

GAUSSIAN 量子化学計算パッケージなどの大規模計算 job は、NQS の qdel コマンドで削除するとしばしば異常終了することがある。我々は、GAUSSIAN job を調べたり、CPU 時間超過の GAUSSIAN job を削除できるより柔軟なスクリプトプログラムを注意深く作成した。このスクリプト使用することによって GAUSSIAN job は正常に終了でき、実際運用試験を行ったところ、異常終了率は 8-12 %の割合で減少するという結果になった。

キーワード

ジョブ運用管理、大規模計算ジョブ、NQS

Discussion and Foundamental Study on the Running and Managing a Huge-sized Computational Job in the Workstation (1).

Yasuo Musashi, Kenichi Sugitani*
Information Processing Centre
2-39-1 Kurokami Kumamoto 860, JAPAN.
TEL +81-96-342-3915 , *+81-96-342-3914 , FAX +81-96-342-3829
musashi@gpo.kumamoto-u.ac.jp, sugitani@eecs.kumamoto-u.ac.jp

Abstract

The huge-sized job process such as GAUSSIAN quantum chemistry packages often does not normally terminate with use of a qdel command of the NQS. We carefully prepared more flexible script programs for checking the GAUSSIAN process and killing the CPU time-overusing that job. Using this scripts, GAUSSIAN job is normally terminated and a real use of running server at our center, the rate of abnormal job termination significantlly decreases by 8-12 %.

Keywords

Running and Managing, huge-sized computational job, NQS

1. 緒言

最近熊本大学総合情報処理センターは並列計算機 (Convex SPP1000 /XA8) をセンター内に設置し、ABAQUS(構造解析) [1] や GAUSSIAN (分子軌道計算)[2] などの大規模数値計算 Job を行える環境を構築している。ABAQUS や GAUSSIAN のような大規模計算 Job は、メモリ及びディスク資源を多量に消費する大規模 Job 系である。SPP1000 は SPP-UX という UNIX 系 OS が動作しており、NQS(Network Queueing System) 使用して Job 管理を行っている。ところが SPP-UX の NQS には CPU 時間超過 Job を削除する機能がない。そこで当センターでは Perl で CPU 時間監視スクリプトを作り、それを時限式にして Job を管理している。しかしながら、Job を削除するために NQS 付属コマンドの qdel を使用したところ、巨大な作業ファイルや core ファイルが残る問題が多発した。このような問題はシステム運用管理者の負荷を増大させるばかりか非生産的である。本研究は UNIX 系 OS を持つシステムにおいて、ABAQUS や GAUSSIAN などの大規模数値演算 Job を安定して動作させ、システム運用管理者にできるだけ負担を掛けない様なシステム環境を構築する方法を研究開発することが目的である。

今回は、大規模 Job である、GAUSSIAN Job の運用の場合について取り上げ、(1) 利用者専用スクリプトによる GAUSSIAN Job の動作状況の把握、(2) CPU 時間超過 Job 監視スクリプトの開発とそのスクリプトを使用する事によって新たに発生した問題点、及び (3) (2) のスクリプト修正後の結果について報告する。

2. 計算方法及び使用言語

プログラム開発は、(1) 利用者スクリプトには C-Shell [3] スクリプト言語を、(2)CPU 時間超過 Job 監視スクリプトには Perl 言語 (perl-4.036)[4] の使用して行った。これらのスクリプトのプログラムコードやアルゴリズムは結果と考察の節で詳細に述べる。

3. 結果と考察

3.1 GAUSSIAN 利用者へ配布したスクリプト

利用者に配布したスクリプトは、以下の通りに記述されている。

```
#!/bin/csh -f
setenv GAUSS_EXEDIR /apl/g94
#setenv GAUSS_SCRDIR /work1
#setenv GAUSS_SCRDIR /work2
#setenv GAUSS_SCRDIR /work3
setenv GAUSS_SCRDIR  /work4
#
setenv GAUSS_SCRDIR1 /work1
setenv GAUSS_SCRDIR2 /work2
setenv GAUSS_SCRDIR3 /work3
setenv GAUSS_SCRDIR4 /work4
#
set IN  = ts.mp2.6-31Gdd
```

```
set DAT = gausWork
#
set DATA = $HOME/$DAT
set IN   = $DATA/$IN
set OUT  = $IN.out
set CT   = `date`
set JID  = `getjid`
echo Gaussian JOB$JID $CT $$ started | memo
#
cd $DATA
touch $OUT
if(-e $OUT) rm -f $OUT
#
(/bin/time /bin/mpa -core 0 $GAUSS_EXEDIR/g94 < $IN )\\
 >& $OUT
#
set GPIDS=`grep PID= $OUT | cut -d= -f3 | tr -d '.'`
foreach GPID($GPIDS)
    set CHKFile=$GAUSS_SCRDIR/g94-$GPID.chk
    set INTFile=$GAUSS_SCRDIR/g94-$GPID.int
    set RWFile=$GAUSS_SCRDIR/g94-$GPID.rwf
    set D2EFile=$GAUSS_SCRDIR/g94-$GPID.d2e
    set SCRFile=$GAUSS_SCRDIR/g94-$GPID.scr
    set CHKFile1=$GAUSS_SCRDIR1/g94-$GPID.chk
    set INTFile1=$GAUSS_SCRDIR1/g94-$GPID.int
    set RWFile1=$GAUSS_SCRDIR1/g94-$GPID.rwf
    set D2EFile1=$GAUSS_SCRDIR1/g94-$GPID.d2e
    set SCRFile1=$GAUSS_SCRDIR1/g94-$GPID.scr
      .
      .
      .
    if(-e $CHKFile)  rm -f $CHKFile
    if(-e $INTFile)  rm -f $INTFile
    if(-e $RWFile)   rm -f $RWFile
    if(-e $D2EFile)  rm -f $D2EFile
    if(-e $SCRFile)  rm -f $SCRFile
    if(-e $CHKFile1) rm -f $CHKFile1
    if(-e $INTFile1) rm -f $INTFile1
    if(-e $RWFile1) rm -f $RWFile1
    if(-e $D2EFile1) rm -f $D2EFile1
    if(-e $SCRFile1) rm -f $SCRFile1
      .
      .
      .
end
set CT   = `date`
echo Gaussian JOB$JID $CT $$ terminated | memo
# END OF GAUSSIAN SCRIPT
```

Table 1: Gaussian Job の投入・終了状況（1995 年 5 月～1996 年 7 月）

	投入件数	正常終了件数	異常終了件数
利用者 1	898	763	135
利用者 2	305	261	44
利用者 3	110	83	27
その他	23	19	4
合計	1336	1126	210

このスクリプトは環境変数の初期化後 GAUSSIAN プロセスを実行し、GAUSSIAN プロセス終了後に作業ファイルの残りを削除して終了するようにコーディングされている。このスクリプトには echo 文部分に getjid と memo という簡単な記録用スクリプトが含ませており、NQS によって実行された GAUSSIAN Job の利用状況が把握できる (利用者 ID や Job の正常・異常終了の記録と取っている)。その記録から Table 1 に示すような結果が得られた。この表から、明らかに 1995 年 5 月から 1996 年 7 月の間に約 16 %の GAUSSIAN Job が異常終了していることが判る (GAUSSIAN Job は正常に終了した場合は作業ファイルを残さないが、実行中の全てのプロセスに同時 SIGKILL を発行すると、作業ファイルを残したまま終了する。この場合を GAUSSIAN が異常終了したと定義する)。GAUSSIAN の異常終了時の巨大な残留ファイルは、システム運用上さまざまな悪影響を及ぼす。この問題を解決するために下記のような find コマンドで深夜に、作成されてから一週間経過した日付を持つファイルやディレクトリを削除するコマンドを時限式で発行するようにした。同時に多くの GAUSSIAN Job が投入されるのでなければ、この方法は有効であると考えたが、

```
0 3 * * * find /work1 -mtime +7 -name "*" -exec rm\\
 -rf {} \;
15 3 * * * find /work2 -mtime +7 -name "*" -exec rm\\
 -rf {} \;
30 3 * * * find /work3 -mtime +7 -name "*" -exec rm\\
 -rf {} \;
45 3 * * * find /work4 -mtime +7 -name "*" -exec rm\\
 -rf {} \;
```

これは GAUSSIAN Job が投入される頻度が週 2,3 本であれば、恐らく有効であると考えられたが、GAUSSIAN Job は続けて投入されている場合が多く、期待するところの効果は得られなかった。

3.2 CPU 超過 Job 監視スクリプト

現在当センターで運用している SPP1000 に搭載された NQS は、CPU 利用時間超過の Job を削除できないという欠陥があるため、なんらかの方法でこれを補う必要がある。これらの理由から、SPP-UX 運用当初から、元 Convex 社の SE の協力で下記に示すようなスクリプトを作成した。

```
#!/usr/local/bin/perl
$dir = "/usr/adm/limit";
#  Define queue names for batch queues
   $a        = "A"; #
   $b        = "B"; # 2.6 hours
   $c        = "C"; # 24 hours
   $d        = "D"; # unlimited
```

```
   $x       = "X"; # unlimited
#  Define cpu limits for batch jobs
   $limit{$a}     =  480;   # 'A' queue
   $limit{$b}     =  9600;  # 'B' queue
   $limit{$c}     =  86400; # 'C' queue
   $limit{$d}     = 0;      # 'D' queue
   $limit{$x}     = 0;      # UNLIMITED QUEUE
# Unbuffer output
$| = 1;
#  Loop through all jobs
open(QSTAT,"/usr/convex/bin/qstat -a| /bin/grep\\
RUNNING | /bin/grep -v type= |");
while(<QSTAT>) {
   s/^ *//;
   #  Extract info from QSTAT output
   ($x,$req0,$reqid,$uname,$pri,$x,$x) = split;
   if ($pri eq 'RUNNING') {
 $uname = $reqid;
 $reqid = $req0;
   }
 $secs = 0;
 $qname = "NULL";
 open(QPS,"/usr/convex/bin/qps -r $reqid |\\
 /bin/grep $reqid|");
 while(<QPS>) {
    s/^ *//;
    #  Extract info from QPS output
    ($qname,$x,$x,$x,$time,$x,$x) = split;
    #  Convert time into seconds
    @tim = split(':', $time);
    $secs = $secs + 60 * $tim[0] + $tim[1];
 }
close(QPS);
if (($secs > 0) && ($qname ne 'NULL') &&\\
 ($limit{$qname} != 0)) {
   if ($secs > $limit{$qname}) {
           system("/usr/convex/bin/qps -r $reqid  | \\
     /bin/cat $dir/CarbonCopy $dir/nqslimit.hd.msg\\
 -   $dir/nqslimit.tl.msg | \\
     /usr/bin/mailx \\
     -s \"CPU TIME LIMIT [$limit{$qname} secs]\\
 EXCEEDED - $reqid\" \\
     $uname");
     system("/usr/convex/bin/qdel -15 $reqid");
           system("/usr/convex/bin/qps -r $reqid|grep \\
 $reqid > /dev/null 2>&1");
     if ($?>>8 == 0) {
            system("/usr/convex/bin/qdel -k $reqid");
     }
```

Table 2: Gaussian Job の投入・終了状況（1996 年 8 月～1997 年 6 月）

	投入件数	正常終了件数	異常終了件数
利用者 1	1614	1447	167
利用者 2	291	237	54
利用者 3	100	83	17
その他	3	3	0
合計	2008	1770	238

```
   }
 }
}
close(QSTAT);
```

このスクリプトは次に述べるような構造をしている。NQS のコマンド群の中の qstat で全 Job の Job プロセス ID を取得し、次に qps コマンドで NQS によって実行されているプロセスの CPU 使用時間を取得して、超過分プロセスがあれば qstat で取得した Job プロセス番号で qdel コマンドを発行し、その旨を該当利用者に E-mail で通知するというものである。このスクリプトは時限式で運用されている。しかしながら qdel コマンドを用いているため、これが原因で GAUSSIAN Job 正常に終了せず、異常終了する可能性がある。これを確かめるために、CPU 時間超過した GAUSSIAN Job を故意に流したままにし、上述のスクリプトを手動で実行したところ、CPU 時間超過 Job プロセスは全て削除されたものの、やはり作業ファイルが残り、GAUSSIAN Job は正常終了しないことが明らかになった。この理由から GAUSSIAN Job は qdel で削除するのではなく、別の方法を採らなければならないことがわかった。

幸いにも当大学は GAUSSIAN をソースファイル形式で購入しているので、プログラムの動作を直接追跡できる。その追跡調査の結果、一つのキープロセスを殺すと、GAUSSIAN の最後に起動されるプロセスが作業ファイル削除などの後処理を行うことが判明したので、このキープロセスだけを上述のスクリプトに優先的に削除させるようにすれば、上述の問題が解決できると考えられる。そこでこの点に注目して CPU 時間超過監視スクリプトを修正を行い、一ヵ月間運用を続けたところ、下記の Table 2 および 3 のような結果が得られた (Table 2 と Table 3 は、それぞれ上述のスクリプトの修正前と修正後の、GAUSSIAN Job のログの結果である)。 修正前では GAUSSIAN Job の異常終了は 12 %であったのに対し修正後は 4 %になったことが判る。これは明らかにスクリプトの修正の効果を示すものである。ただし修正後あまり時間が経過しておらず、そのデータの信頼性が高くはないと考えられるが、当センターではこの調査結果をベースにシステム運用管理基礎技術をより高度なものにしていけるものと考えている。

Table 3: Gaussian Job の投入・終了状況（1997 年 7 月）

	投入件数	正常終了件数	異常終了件数
利用者 1	158	149	9
利用者 2	53	53	0
利用者 3	6	6	0
その他	0	0	0
合計	217	208	9

4. 結論

GAUSSIAN 利用を把握するため、専用の利用者スクリプトを作成した。それから得られる GAUSSIAN Job の異常終了率は、約 12-16 % であることが示された。SPP-UX の NQS には CPU 時間超過 Job を kill する機能がないため、Perl で CPU 時間超過削除スクリプトを作り、更に GASSIAN Job を差別的に扱い、その Job から発行されたキープロセスを優先的削除することにより、異常終了が 4 %までに減少する効果が得られた。

5. 謝辞

すべての計算およびスクリプトプログラム開発は熊本大学総合情報処理センターにおける HP Exemplar SPP1000/XA8 scalable parallel processor で行った。このプログラム開発に当たり、元 Convex 社の稲美氏及び現 HP 社の松尾氏には大変お世話になったのでここで感謝の意を表す。

6. 参考文献

[1] ABAQUS/Standard version 5.6, Hibbitt, Karlsson and Sorensen, Inc., 1996.

[2] M. J. Frisch, G. W. Trucks, H. B. Schlegel, P. M. W. Gill, B. G. Johnson, M. A. Robb, J. R. Cheeseman, T. A. Keith, G. A. Petersson, J. A. Montgomery, K. Raghavachari, M. A. Al-Laham, V. G. Zakrzewski, J. V. Ortiz, J. B. Foresman, J. Cioslowski, B. B. Stefanov, A. Nanayakkara, M. Challacombe, C. Y. Peng, P. Y. Ayala, W. Chen, M. W. Wong, J. L. Andres, E. S. Replogle, R. Gomperts, R. L. Martin, D. J. Fox, J. S. Binkley, D. J. Defrees, J. Baker, J. P. Stewart, M. Head-Gordon, C. Gonzalez and J. A. Pople, GAUSSIAN 94, Gaussian Inc., Pittsburg PA, 1995.

[3] G. Aderson and P. Aderson, in : UNIX C SHELL FIELD GUIDE, Pretince-Hall Inc., 1986.

[4] L. Wall and R. L. Schwarz, in : Programming PERL, O'Reilly and Associates, Inc., 1995.