UNIX系OSにおける計算ジョブ運用管理に関する基礎研究

武蔵　泰雄*1、朱　紅兵*2、杉谷 賢一*3

熊本大学総合情報処理センター
〒860-8555　熊本市黒髪2-39-1
*1TEL：096-342-3915
FAX：096-342-3829
musashi@gpo.kumamoto-u.ac.jp

*2TEL：096-342-3912
FAX：096-342-3829
shu@eecs.kumamoto-u.ac.jp

*3TEL：096-342-3914
FAX：096-342-3829
sugitani@eecs.kumamoto-u.ac.jp

概要

研究室単位のワークステーションを計算実験サーバとして複数の利用者で使用する場合は、計算実験ジョブを整理して順番に実行して行くジョブ管理システムが必要である。我々は簡単な計算実験ジョブ管理制御スクリプトプログラムを作り、これを用いてジョブ制御を複数の一般利用者で制御する場合と、単一制御専用利用者に集中させた 2 つ実験を行い、後者が優れていることを見出した。

キーワード

ジョブ制御、計算実験、スケジューリング

A Foundamental Study on Managing Calculation Job in UNIX-like Operating System.

Yasuo Musashi*[1], Hongbing Zhu*[2], Kenichi Sugitani*[3]

Information Processing Center, 2-39-1 Kurokami Kumamoto 860-8550, JAPAN.
*[1]TEL：+81-96-342-3915 FAX：+81-96-342-3829 musashi@gpo.kumamoto-u.ac.jp
*[2]TEL：+81-96-342-3915 FAX：+81-96-342-3829 shu@eecs.kumamoto-u.ac.jp
*[3]TEL：+81-96-342-3914 FAX：+81-96-342-3829 sugitani@eecs.kumamoto-u.ac.jp

Abstract

In the case of using workstation in the laboratory as a calculation server by several persons, we must have the system which helps managing calculation jobs to sort submitted and numbered jobs in order, and to execute these jobs. We prepared the simple shell-script program which controls the calculation job. We performed two experiments: the former is controlled by several general users but the latter is controlled by only one user specialized only managing job. The results show that the latter is favorable for job managing system.

Keywords

Job managing, experimental calculation, scheduling

1．緒言

去年の前半、大学の研究室ではワークステーション（WS）が計算実験サーバであることが多かったが、最近PCの高性能化が進み、Linux等に代表されるPC-UNIXの拡がりつつあることは良く知られている。この動きに連動して一部のアプリケーションソフトウェアメーカーはLinux/FreeBSD対応版の開発を進めており、将来PC-UNIXがアプリサーバとしての地位を確立するかも知れない。実際周囲の大学の研究室等では現実にこの現象が起きている。PCだったらあまり高価ではなく、UNIX系OSが動き、しかも最近のPCは一世代前WSとより高速である、などの利点が多く、従ってPC-UNIXが急速に普及する可能性が高い[1]。

しかしながら、PC-UNIXであろうとWSであろうとUNIX環境で計算実験を行う場合複数の利用者で使うのが普通である。サーバの数や利用者が多くなれば、協調的雰囲気の中で計算実験を続けることは時と場合によっては難題に直面する。難題とはなる点は、計算時間の予想が付

かないジョブがありスケジュールがうまく組めない、一利用者が何本もジョブを走らせる者がいて一部の研究を阻害したりするなどである。しかるに利用者間の感情を害せず、ジョブを投入順に整理して流すシステム、すなわちジョブ管理制御システムが必要がある。幸い PC-UNIX については市販させているジョブ管理制御ソフトがあるが、研究室で台数分だけライセンスを取得するためにはやはり高価である[2]。

そこで我々は UNIX 汎用の実験用ジョブ管理制御を作成し、ジョブ管理ソフトに最低必要な機能、ジョブ制御行列（キュー）の管理・制御方法について調査・研究したので報告する。

2．計算方法

2．1　制御行列（キュー）

キューを下記の規則に従って特定ディレクトリに作成する。

- 実行識別子: 実行中であることを示すファイルで、第一文字が j ではじまり、次に 6 桁の数字文字列、8 文字目が e である。ファイル内容はジョブ投入者の、ログイン ID、グループ ID 及び、実行中のスクリプトが記録されてる。例: j000001e, j000010e
- 待ち識別子: 投入後待ちであることを示すファイルで、第一文字が j ではじまり、次に 6 桁の数字文字列、8 文字目が w である。ファイル内容はジョブ投入者の、ログイン ID、グループ ID 及び、実行するスクリプトが記録されてる。例: j000001w, j000010w
- 削除識別子: キャンセル予定のジョブ番号を示すファイルで、第一文字が j ではじまり、次に 6 桁の数字文字列、8 文字目が c である。ファイル内容はジョブ投入者の、ログイン ID、グループ ID 及び、実行中または実行予定のスクリプトが入っている。例: j000001c,j000010c

2．2　キュー制御原理

キュー制御原理は次の通りである。最初にキューから実行キューを抜き出す。実行キューはその数に制限があり、これを越えることは制御がうまくできていないことを示す。実行キューの数が制限数より小さければ、待ち行列の最前列に位置する待ち識別子を調べそれを mv コマンドで実行識別子に変更し、その記録されている、実行すべきスクリプトをログイン ID、ユーザー ID で実行する。最後に削除識別子をチェックし、待ち識別子に該当のジョブがあれば該当の待ち識別子そのものを削除し、実行識別子にあれば、実行スクリプトのプロセス ID（PID）とその子プロセス ID を調べだしすべてのプロセスに SIGKILL シグナルを発行して、強制終了させ、最後に該当の実行識別子を消す。これらの作業を一定時間 sleep した後また繰り返す。以下にその簡単スクリプトを C シェル[3]を用いて記述した。

```
#!/bin/csh
if（$#argv = = 0）　set argv = 60
while（1）
```

```
set EMATX=`ls $QUEUE | grep 'j.....e'`
switch ($#EMATX)
    case 0:
     set WMATX=`ls $QUEUE | grep 'j.....w'`
     set JPID= `ls $QUEUE | grep 'j.....w' | cut -c2-7`
     set LGIN= cat `ls $QUEUE | grep 'j.....w' | cut -d' ' -f1`
     set GID = cat `ls $QUEUE | grep 'j.....w' | cut -d' ' -f2`
     set USPT= cat `ls $QUEUE | grep 'j.....w' | cut -d' ' -f3`
     set GOIN1= j"$JPID[1]"e
     set GOIN2= j"$JPID[2]"e
     mv   $QUEUE/$WMATX[1] $QUEUE/$GOIN1
     Go   $LGIN[1] $GID[1] $USPT[1] &
     mv   $QUEUE/$WMATX[2] $QUEUE/$GOIN2
     Go   $LGIN[2] $GID[2] $USPT[2] &
     breaksw
    case 1:
     set WMATX=`ls $QUEUE | grep 'j.....w'`
     set JPID= `ls $QUEUE | grep 'j.....w' | cut -c2-7`
     set LGIN= cat `ls $QUEUE | grep 'j.....w' | cut -d' ' -f1`
     set GID = cat `ls $QUEUE | grep 'j.....w' | cut -d' ' -f2`
     set USPT= cat `ls $QUEUE | grep 'j.....w' | cut -d' ' -f3`
     set GOIN= j"$JPID[1]"e
     mv   $QUEUE/$WMATX[1] $QUEUE/$GOIN
     Go   $LGIN[1] $GID[1] $USPT[1] &
     breaksw
         breaksw
     default:
         breaksw
     endsw
     set CANJOB=`ls $QUEUE | grep 'j.....c' | cut -c2-7`
     if  ( $#CANJOB == 0 )  goto CanEnd
     foreach CANC ($CANJOB)
         set CANNUM = 'j'$CANC'e'
         set CANE=`ls $QUEUE | grep $CANNUM`
         set GPID=`/usr/local/bin/getCanPID $CANNUM`
         kill -9 $GPID[$#GPID]
         rm -f $QUEUE/j"$CANC"e
         rm -f $QUEUE/j"$CANC"c
     end
     CanEnd:
```

```
    sleep $1
end
exit
```

3．実験方法

ジョブの管理制御実験について下記の 2 つ方法を用いて行った。

(1)　ジョブ投入ごとに制御ソフト実行する。

(2)　一人の利用者ログイン ID で集中管理する。

実験に使用したシステムは、Linux-2.0.33 + Intel MMX 200MHz、Physical Memory 160MB、EIDE 3.2GB、 Gaussian 94（e2） [5]であり、C シェル[3]、C 言語[4]を使用してプログラムの開発を行った。

4．結果と考察

4．1　投入時ごとに制御ソフトを実行する

ジョブを投入するごとにキューの制御ソフトを起動する（図 1A）。この方法では、すべてのプロセス ID が投入者の所有になるという便利さがある。しかしながら、一利用者で管理する場合は、利用者のジョブスクリプト投入時にログイン ID とグループ ID をどこかに保存し、実行時にスクリプトの PID の所有者を、ジョブ投入者のログイン ID とグループ ID に変更して実行する必要がある。

最初にこのジョブ投入と同時に制御を投入者の所有で行ったところ実行ジョブが決まった数へ固定できないという不具合が見付かった。調査を続けたところ、sleep コマンドで制御スクリプトをサスペンドさせているが、サスペンドから復帰が同時に起こるようにしたところ、制御スクリプトが実行ジョブが規定数より少ないと判断し、ジョブを制限数よりも多く実行されることが分かった。

そこで乱数を使って複数流れているキューの制御スクリプトの sleep に与える引数を散らし、同時にサスペンドから復帰しないようにしたが、数字間で同時に復帰することが起こってしまった。これはアプリケーションの終了時刻によって一致してしまうことがあるためである。従って複数利用者でキューを制御スクリプトを走らせる場合は問題があることが示された。

4．2 一利用者に集中してキューを制御させる

キュー制御専用の利用者を作成し、キューを制御させる実験を行った（図 1B）。現在のところ問題は発生していない。ただし、他の利用者のスクリプトをその利用者のプロセス ID で実行するためには、その利用者のホームディレクトリを other まで読みだし可にしなければならない。これは多少セキュリティ的に問題であるが、研究室レベルであれば、利用者のセキュリティ意識を高めることで回避できると考える。

5．結論

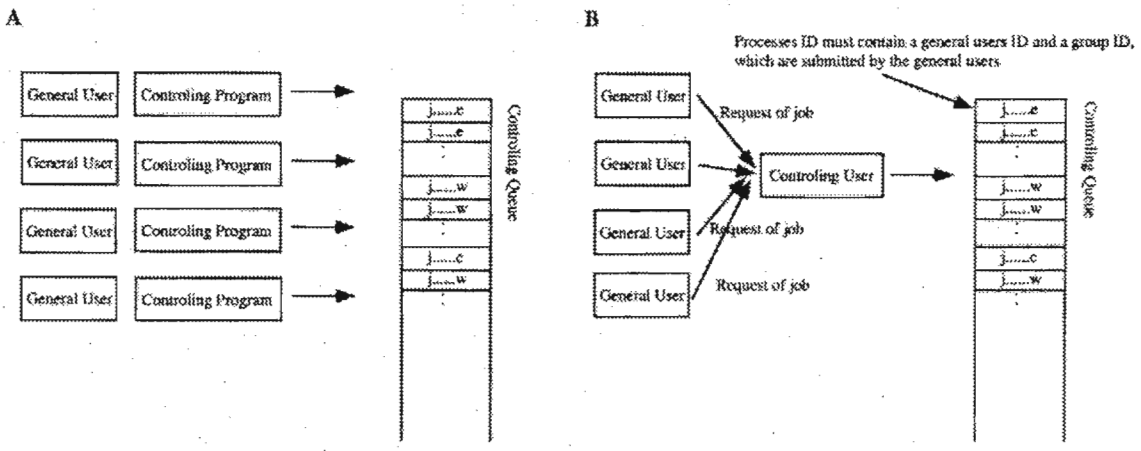
PC-UNIX 上で計算実験ジョブを投入順に整理して実行するジョブ管理制御システムを簡単なジョブ制御行列を使用して作成した。このジョブ制御プログラムを用いて、投入者自身の制御の場合と制御管理専用利用者によるジョブ制御の場合の 2 組み実験を行ったところ、複数の利用者による制御よりも、単一の専用利用者を作成して集中管理による制御が安定して稼働することが示された。

6．謝辞

すべての実験およびプログラム開発は熊本大学総合情報処理センターにおいて行った。また本プログラムで初期開発で協力していただいた工学部修士課程の峯君、時松君及び小川君（当時 M2）には大変お世話になったのでここで感謝の意を表します。

7．参考文献

[1] 山崎康宏 and はねひでや, in : Linux パワーガイドブック,技術評論社, 1997.
[2] Load Sharing Facility, Platform Computing Co., 1992.
[3] G. Aderson and P. Aderson, in : UNIX C SHELL FIELD GUIDE, Pretince-Hall Inc., 1986.
[4] gcc-2.7.2.1, Free Software Foudation, Inc., 1996.
[5] GAUSSIAN 94 Rev. e2, Gaussian Inc., Pittsburg PA, 1995.


A
General User
Controling Program
General User
Controling Program
General User
Controling Program
General User
Controling Program
Controlling Queue
B
Processes ID must contain a general users ID and a group ID, which are submitted by the general users
General User
General User
General User
General User
Request of job
Request of job
Request of job
Controling User
Controlling Queue