

Matrix Multiplication with POP

Performance analysis of a distributed matrix multiplication program

Alshweiki Mhd Ali ¹
Gugger Joël ²
Marguet Steve-David ³

user: ggroup20@grid11

May 7, 2016

¹ mhdali.alshweiki@master.hes-so.ch

² joel.gugger@master.hes-so.ch

³ stevedavid.marguet@master.hes-so.ch

Abstract

The objective of this lab is to execute and to analyse the performances of a parallel square matrices multiplication program written in POP-C++ and in POP-Java. As for the MPI/OpenMP lab, these programs computes square matrices multiplication, i.e. the product $A \times B = R$ where A , B and R are $N \times N$ matrices (square matrix).

The program uses a « Master/Worker » approach. The master prepares the matrices, creates the workers (POP-C++ or POP-Java parallel objects), sends the work to do to each workers, waits for the partial result of each worker and finally reconstructs the R matrix.

The algorithm behaves similarly to the one of the MPI/OpenMP lab by dividing the matrix A in several bloc of lines and the matrix B in several blocs of columns.

Chapter 1

Computation of sequential references times

The sequential reference time is the time used to do the computation using only one worker and one core.

Chapter 2

Computation of parallel times

Each group will have to compute for five different sizes of the matrix (N), the time for five different numbers of workers (W). Our group will made computation for this sizes:

Matrix sizes (N)
1080
2160
3240
4620
6240

Workers (W)	$= LxC$
2	$= 1x2$
4	$= 2x2$
6	$= 2x3$
9	$= 3x3$
10	$= 5x2$

We had some difficulties to execute our script correctly. The first time, the script haven't be executed because relative path. The second time, the script ran but we compute only 25 calculs. We have made only one size by worker size.

Listing 2.1: Cron job

```

1  # Edit this file to introduce tasks to be run by cron.
2  #
3  # For example, you can run a backup of all your user accounts
4  # at 5 a.m every week with:
5  # 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
6  #
7  # For more information see the manual pages of crontab(5) and cron(8)
8  #
9  # m h dom mon dow   command
10 SHELL=/bin/bash
11 10 0 29 4 * /etuhome/ggroup20/project/POPC/MATRIX/runme.sh 2>&1 > /etuhome/ggroup20
    /project/POPC/MATRIX/cronlog.log
12 10 17 2 5 * /etuhome/ggroup20/project/POPC/MATRIX/minirun.sh 2>&1 >> /etuhome/
    ggroup20/project/POPC/MATRIX/cronlog.log
13 10 12 3 5 * /etuhome/ggroup20/project/POPC/MATRIX/runme.sh 2>&1 >> /etuhome/
    ggroup20/project/POPC/MATRIX/cronlog.log
14 10 13 6 5 * /etuhome/ggroup20/project/POPC/MATRIX/runme.sh 2>&1 >> /etuhome/
    ggroup20/project/POPC/MATRIX/cronlog_125.log
15 10 15 6 5 * /etuhome/ggroup20/project/POPC/MATRIX/runme.sh 2>&1 >> /etuhome/
    ggroup20/project/POPC/MATRIX/cronlog_125_1_1.log
16 10 16 6 5 * /etuhome/ggroup20/project/POPC/MATRIX/runme.sh 2>&1 >> /etuhome/
    ggroup20/project/POPC/MATRIX/cronlog_125_1_1.log

```

This version is the third version that execute the 100 calculation we missed. This why in each size we have a commented line.

Listing 2.2: Final bash script

```

1  #! /bin/bash
2
3  export POPC_LOCATION=/opt/popc/
4  export PATH=${PATH}:${POPC_LOCATION}/bin:${POPC_LOCATION}/sbin
5
6  DATE=$(date)
7
8  echo "$DATE"
9  echo "$PATH"
10
11 cd /etuhome/gggroup20/project/POPC/MATRIX/
12 pwd
13
14
15 # TODO
16 #|w  |= L x C| size=N|
17 #|--|-----|-----|
18 #|2  |= 1 x 2| 1080 |
19 #|4  |= 2 x 2| 2160 |
20 #|6  |= 2 x 3| 3240 |
21 #|9  |= 3 x 3| 4620 |
22 #|10 |= 5 x 2| 6240 |
23
24 # Cleanup an rebuild everything
25 #make clean && make all;
26
27 # Copy the machine name in machines.txt
28 cat ./machines_cores.txt > ./machines.txt
29
30 # Lets gets started
31 touch ./output_125.log
32 OUT=./output_125.log
33
34 echo "$DATE" >> $OUT;
35
36 for size in 6240 4620 3240 2160 1080
37 do
38     for i in {1..5}
39     do
40         case $size in
41             1080 )
42                 #echo $(date) >> $OUT;
43                 #popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
44                 echo $(date) >> $OUT;
45                 popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
46                 echo $(date) >> $OUT;
47                 popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
48                 echo $(date) >> $OUT;
49                 popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
50                 echo $(date) >> $OUT;
51                 popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
52             ;;
53             2160 )

```

```

54         echo $(date) >> $OUT;
55         popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
56         #echo $(date) >> $OUT;
57         #popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
58         echo $(date) >> $OUT;
59         popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
60         echo $(date) >> $OUT;
61         popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
62         echo $(date) >> $OUT;
63         popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
64     ;;
65     3240 )
66         echo $(date) >> $OUT;
67         popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
68         echo $(date) >> $OUT;
69         popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
70         #echo $(date) >> $OUT;
71         #popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
72         echo $(date) >> $OUT;
73         popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
74         echo $(date) >> $OUT;
75         popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
76     ;;
77     4620 )
78         echo $(date) >> $OUT;
79         popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
80         echo $(date) >> $OUT;
81         popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
82         echo $(date) >> $OUT;
83         popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
84         #echo $(date) >> $OUT;
85         #popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
86         echo $(date) >> $OUT;
87         popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
88         echo $(date) >> $OUT;
89     ;;
90     6240 )
91         echo $(date) >> $OUT;
92         popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
93         echo $(date) >> $OUT;
94         popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
95         echo $(date) >> $OUT;
96         popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
97         echo $(date) >> $OUT;
98         popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
99         #echo $(date) >> $OUT;
100        #popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
101    ;;
102    esac
103    done
104    done

```

Abstract

Les sources du projet sont disponibles sur GitHub à l'adresse suivante :
<https://github.com/Alshweiki/ProgAlg-Lab2>