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.

Listing 1.1: Sequential results

```
Fri May  6 15:10:01 CEST 2016
Fri May 6 15:10:01 CEST 2016
6240    1       1       0.71077 5.55884 239.229 4       5.82293 235.667
Fri May 6 15:14:07 CEST 2016
6240    1       1       0.656745        5.54313 239.3   4       5.81544 235.721
Fri May 6 15:18:14 CEST 2016
6240    1       1       0.674083        5.54691 239.358 4       5.81711 235.807
Fri May 6 15:22:20 CEST 2016
4620    1       1       0.672018        3.0396  97.6173 4       3.19584 95.6442
Fri May 6 15:24:02 CEST 2016
4620    1       1       0.660868        3.03455 97.5435 4       3.19838 95.5594
Fri May 6 15:25:44 CEST 2016
4620    1       1       0.665649        3.03751 97.5537 4       3.19701 95.5854
Fri May 6 15:27:26 CEST 2016
3240    1       1       0.653154        1.48841 34.5118 4       1.5814  33.5181
Fri May 6 15:28:03 CEST 2016
3240    1       1       0.647843        1.47855 34.0019 4       1.5793  32.9971
Fri May 6 15:28:40 CEST 2016
3240    1       1       0.649754        1.47739 33.9203 4       1.58107 32.9127
Fri May 6 15:29:16 CEST 2016
2160    1       1       0.636022        0.654782        10.2599 4       0.722186        9.7841
Fri May 6 15:29:28 CEST 2016
2160    1       1       0.633916        0.645487        10.2445 4       0.712177        9.76236
Fri May 6 15:29:40 CEST 2016
2160    1       1       0.625118        0.650578        10.2606 4       0.715434        9.78311
Fri May 6 15:29:51 CEST 2016
1080    1       1       0.649178        0.162991        1.69181 4       0.186798        1.5421
Fri May 6 15:29:54 CEST 2016
1080    1       1       0.636671        0.166621        1.40267 4       0.195391        1.2522
Fri May 6 15:29:56 CEST 2016
1080    1       1       0.654924        0.169804        1.39129 4       0.19328 1.2465
Fri May  6 16:10:01 CEST 2016
Fri May 6 16:10:01 CEST 2016
6240    1       1       0.720963        5.55943 239.468 4       5.8245  235.902
Fri May 6 16:14:08 CEST 2016
6240    1       1       0.66236 5.55007 239.563 4       5.82148 235.988
Fri May 6 16:18:14 CEST 2016
6240    1       1       0.66772 5.55392 239.211 4       5.81839 235.645
Fri May 6 16:22:21 CEST 2016
4620    1       1       0.671772        3.03342 97.75   4       3.20014 95.768
Fri May 6 16:24:03 CEST 2016
4620    1       1       0.668514        3.03641 97.4617 4       3.19583 95.4944
```

```
Fri May 6 16:25:45 CEST 2016
4620    1       1       0.666675        3.03827 97.6856 4       3.20809 95.707
Fri May 6 16:27:27 CEST 2016
3240    1       1       0.659824        1.48298 33.9517 4       1.58078 32.9525
Fri May 6 16:28:03 CEST 2016
3240    1       1       0.6377  1.49207 34.0494 4       1.58924 33.0538
Fri May 6 16:28:40 CEST 2016
3240    1       1       0.653028        1.48943 34.0213 4       1.58381 33.0161
Fri May 6 16:29:17 CEST 2016
2160    1       1       0.623649        0.657485        10.2584 4       0.716694        9.78921
Fri May 6 16:29:28 CEST 2016
2160    1       1       0.644504        0.638023        10.2626 4       0.705568        9.77762
Fri May 6 16:29:40 CEST 2016
2160    1       1       0.662751        0.646143        10.7575 4       0.710278        10.2813
Fri May 6 16:29:52 CEST 2016
1080    1       1       0.638768        0.167578        1.40879 4       0.194913        1.26278
Fri May 6 16:29:55 CEST 2016
1080    1       1       0.642096        0.167531        1.38331 4       0.194629        1.24018
Fri May 6 16:29:57 CEST 2016
1080    1       1       0.632144        0.17172 1.37969 4       0.192929        1.24201
```

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

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

```bash
#! /bin/bash

export POPC_LOCATION=/opt/popc/
export PATH=${PATH}:$POPC_LOCATION/bin:$POPC_LOCATION/sbin

DATE=$(date)

echo "$DATE"
echo "$PATH"

cd /etuhome/ggroup20/project/POPC/MATRIX/
pwd


# TODO
#|w  |= L x C| size=N|
#|:--|-------|------:|
#|2  |= 1 x 2|  1080 |
#|4  |= 2 x 2|  2160 |
#|6  |= 2 x 3|  3240 |
#|9  |= 3 x 3|  4620 |
#|10 |= 5 x 2|  6240 |

# Cleanup an rebuild everything
#make clean && make all;

# Copy the machine name in machines.txt
cat ./machines_cores.txt > ./machines.txt

# Lets gets started
touch ./output_125.log
OUT=./output_125.log

echo "$DATE" >> $OUT;

for size  in  6240 4620 3240 2160 1080
do
    for i in {1..5}
    do
        case  $size in
            1080 )
                #echo $(date) >> $OUT;
                #popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
            ;;
            2160 )
```

```
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
                #echo $(date) >> $OUT;
                #popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
            ;;
            3240 )
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
                #echo $(date) >> $OUT;
                #popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
            ;;
            4620 )
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
                #echo $(date) >> $OUT;
                #popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
                echo $(date) >> $OUT;
            ;;
            6240 )
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 1 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 2 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 2 3 $OUT;
                echo $(date) >> $OUT;
                popcrun ./obj.map ./mainpopc $size 3 3 $OUT;
                #echo $(date) >> $OUT;
                #popcrun ./obj.map ./mainpopc $size 5 2 $OUT;
            ;;
        esac
    done
done
```

Abstract

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