COMP-40730 HPC

Report for Assignment 3

Author:	Paula Dwan
Due date:	13-July-2014
Lecturer:	Alexey Lastovetsky
Subject:	COMP-40730 High Performance Computing
College:	University College Dublin

Contents

EXERCISE

Write a parallel OpenMP program computing the norm of the product of two n×n dense matrices on a p-processor SMP so that

- p threads are involved in the parallel computations.
- The 1-dimensional parallel algorithm of matrix multiplication is employed:
  - one of matrices is partitioned in one dimension into p equal slices
  - there is one-to-one mapping between the partitions and threads
  - each thread is responsible for computation of the corresponding slice of the resulting matrix
- Computation of the norm of the resulting matrix employs the mutex synchronization mechanism.

You can use BLAS or ATLAS for local computations.

Experiment with the program and build:

- The dependence of the execution time of the program on the matrix size n.
- The speedup over a serial counterpart of the program.

Explain the results.

Variants of the assignment:

1. Granularity of the program:
   (a) Two successive steps:
      i. Parallel matrix multiplication
      ii. Parallel computation of the norm of the resulting matrix
   (b) One-step algorithm. No intermediate resulting matrix.

2. Partitioning scheme:
   (a) Left matrix is horizontally partitioned
   (b) Right matrix is vertically partitioned

3. Matrix norm to be computed:
   (a) The maximum absolute column sum norm (aka one-norm):
   (b) The maximum absolute row sum norm (aka infinity-norm):

$$||A||_\infty = \max_{0 \le i < n} \sum_{j=0}^{n-1} |a_{ij}|$$

Overview of Computations Obtained and How

Assignment 3 basically involved (for me) writing one program which utilized open MP when calculating manually and BLAS when calculating otherwise. :

A3-omp.-1D.c

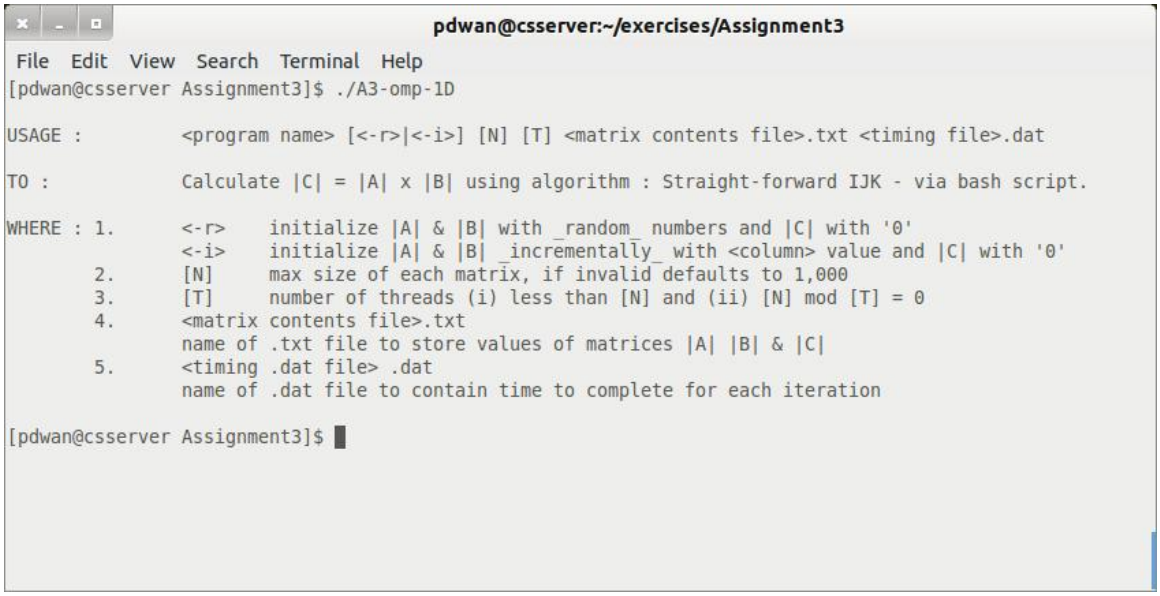
Algorithm

using additional variable

```
for (i=0; i<rows; i++)
{
    for (j=0; j<cols; j++)
    {
        double sum = 0.0;
        for (k=0; k<rows; k++)
        {
            sum+= (A [i][k]) * (B [k][j]);
        }
        C [i][j] = sum;
    }
}
```

|C| matrix was calculated using the algorithm as covered in the first assignment. The same matrix computation was implemented using cblas. Time taken to calculate |C| and the infinity norm was noted and graphed.

This was completed using **A3-omp-1D.c :**



This is compiled using *gcc* and *openmp*:

```
[pdwan@csserver Assignment3]$  gcc -I/home/cs/khasanov/libs/CBLAS/src A3-omp-1D.c -o A3-omp-1D
/home/cs/khasanov/libs/cblas_LINUX.a  /usr/lib/libblas.a -lgfortran -fopenmp
```

and executed on a once off using (for example)

```
[pdwan@csserver Assignment3]$ ./A3-omp-1D -r 10 2 matrix2.txt timings2.dat
```

Where ***-r*** indicates random number generation (for 1 to 10) and ***-i*** indicates number of column + 1.

Assignment Execution

Each program was executed multiple times using the script *./runAssignment3.sh*. This has multiple options and the syntax and usage follows:

```
UCD - Assigments
File  Edit  View  Search  Terminal  Help
✔ (master)~/gitrepos/UCD/COMP40730.HPC/Assignment3$ ./runAssignment3.sh

USAGE : ./runAssignment3.sh \
         -d2|--cblas -r|--random -i|--increment -m|--matrix <n> -t|--thread <t> -v|--values -?|-h|--help

TO :    Calculate |C| = |A| x |B| and then infintiy norm using Open MP

LOGS :  Created in current dir and moved to <logDir> :
        <file>.txt :    matrix values for matrices |A| |B| & |C|,
        <file>.dat :    timing data for each computation
        <file>.log :    summary of stdout.

WHERE : -d2|--cblas     Compile .c source files using dgemm cblas & openmp
        -r|--random     Initialize |A| & |B| with random numbers and |C| with '0'
        -i|--increment  Initialize |A| & |B| incrementally with <column> value and |C| with '0'
                        '-i|--increment' & '-r|--random' are mutually exclusive

        -m|--matrix <n> Matrix dimension, if odd number +1 added or if invalid set to [ 1000 ], thread count set to [ 20 ]
        -t|--thread <t> number of threads, if invalid set to  [ 20 ]  and matrix size set to [ 1000 ]
        -v|--values     Use predefined range of valid values for <nx> and <nb> as follows :
                        <matrixArray> : { 50, 50, 50, 100, 100, 100, 500, 500, 500, 1000, 1000, 1000 }
                        <threadArray> : { 10, 10, 10, 10, 10, 10, 20, 20, 20, 20, 20, 20 }
                        '-m|--matrix <n>' | -t|--thread <t>' and '-v|--values' mutually exclusive.

        -?|-h|--help    usage

✔ (master)~/gitrepos/UCD/COMP40730.HPC/Assignment3$
```

Execute this script in the home directory of Assignment 3.

Note: Please retain the overall directory structure when unzipping.

Note that the script `./runAssignment3.sh` allows two types of implementation

- Multiple iteration : use the switch <-v|--values>, when a predefined range applies for [N] : matrix size and [T] : number of threads applicable.
- Single iteration : use the switch <-m|--matrix> [N] where the user specifies the values for [N] : matrix size and [T] : number of threads applicable.

Running A3-omp-1D *: standalone*

The compiled .c program may also be run standalone. Usage follows:

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help
[pdwan@csserver Assignment3]$ ./A3-omp-1D

ERROR :         <number of arguments> : 1, is invalid, less than <default> : 6.

USAGE :         <program name> [<-r>|<-i>] [N] [T] <matrix contents file>.txt <timing file>.dat

TO :            Calculate |C| = |A| x |B| using Open MP and also calculate infinity norm of |C|

WHERE : 1.      <-r>    initialize |A| & |B| with _random_ numbers and |C| with '0'
                <-i>    initialize |A| & |B| _incrementally_ with <column> value and |C| with '0'
        2.      [N]     max size of each matrix, if invalid defaults to 1,000
        3.      [T]     number of threads (i) less than [N] and (ii) [N] mod [T] = 0
        4.      <matrix contents file>.txt
                name of .txt file to store values of matrices |A| |B| & |C|
        5.      <timing .dat file> .dat
                name of timing data file to containing calculation time for each iteration

[pdwan@csserver Assignment3]$
```

Log Files Obtained

Data text files suitable containing the values of the computation used for matrices |A| and |B| and the results stored in |C| are saved in the appropriate log files. File naming convention via the script is :

<data log file name>	`pdwan-<time>-values-<A3-omp-1D->-<iteration>.txt`
example:	`pdwan-20140708.015835-values-A3-omp-1D-8.txt`

Single iteration also applies where the user enters arbitrary, valid values for matrix size and does not use the scripts and the other required parameters.

A summary file containing processing time for each computation (manual and BLAS) for is also saved. This is in a format suitable for us with GNUplot.

<timing log file name>	`pdwan-<time>-timing-<A3-omp-1D->-<iteration>.dat`
example:	`pdwan-20140708.015835-timing-A3-omp-1D-8.dat`

Finally a log file containing <stdout> for the script was created.

If ***./A3-omp-1D*** is used without the script then files may be named whatever the user wishes and no .log file applies.

Sample timings file :

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help
#  ------------------------------------------------------------------------------------------
#
# Program :     A3-omp-1D
# where :       .dat contains timing data & .txt contains matrix values
#  ------------------------------------------------------------------------------------------
# |Matrix|      |Threads|       Time/manual     Inf Norm/manual         Time/dgemm      Inf Norm/dgemm
#
6       3       0.001521s       441     0.001662s       441
6       3       0.001246s       1396    0.001480s       1396
                                                                                    1,1           Top
```

Sample Matrix values file:

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help
#  ------------------------------------------------------------------------------------------
#
# RUNNING :     ./A3-omp-1D -r 4 2  t7.dat
# Program :     A3-omp-1D
# where :       .dat contains timing data & .txt contains matrix values
#
# Summary of values added to each matrix - retained for later reference and validation
#
#  ------------------------------------------------------------------------------------------


# Initialize results <4> x <4> |A| ...
4       7       8       6
4       6       7       3
10      2       3       8
1       10      4       7

# Initialize results <4> x <4> |B| ...
1       7       3       7
2       9       8       10
3       1       3       4
8       6       10      3

# Initialize results <4> x <4> |C| ...
0       0       0       0
0       0       0       0
0       0       0       0
0       0       0       0

# RESULTS : calculation where number of threads are : 2

# |C| : <4> x <4> matrix computed values : MANUAL ...
90      135     152     148
61      107     111     125
87      139     135     126
89      143     165     144
                                                                                    1,1           Top
```

I wished to keep each .c program as clean as possible and so all production setup was completed in the script for each assignment. Thus file creation and validation for each iteration was completed before the .c program was even called. Simple validation of the arguments passed to each .c program is also completed.

I also spot-checked the results as practical. Results obtained are detailed in Appendix I – Validate Results.

GNUplot Execution

I followed the same convention for each .dat file as produced, an example follows :

Sample .dat file

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help
# ------------------------------------------------------------------------------------------
#
# Program :     A3-omp-1D
# where :       .dat contains timing data & .txt contains matrix values
# ------------------------------------------------------------------------------------------
# |Matrix|      |Threads|       Time/manual     Inf Norm/manual         Time/dgemm      Inf Norm/dgemm
#
6       3       0.001521s       441     0.001662s       441
6       3       0.001246s       1396    0.001480s       1396
                                                                                  1,1           Top
```

Each was then presented in graphical format using GNUplot, comparing times taken for manual and for BLAS/ATLAS computations. A generic GNUplot program was written to output the data to the screen.

GNUplot program execution

```
# To execute, launch GNUplot and run :
# gnuplot> load <filename.gp>
# making sure that the data file name used is updated if needed.

# Paula Dwan : Assignment 3 : A3-plotgraph-matrix.gp
reset
set xtic auto
set ytic auto
set size 1,1
set grid
set key outside
#
set title 'Comparison : Matrix Size v time taken'
set ylabel 'Time taken / s'
set xlabel 'Matrix size'
set origin 0,0
plot 'logDir/pdwan-20140714.051601-data-A3-omp-1D.dat' u 1:3 t 'simple' w l
lw 0.5 lc rgb 'blue', 'logDir/pdwan-20140714.051601-data-A3-omp-1D.dat' u
1:5 t 'dgemm' w l lw 0.5 lc rgb 'red'
#
pause -1

# Paula Dwan : Assignment 3 : A3-plotgraph-thread.gp
reset
set xtic auto
set ytic auto
set size 1,1
set grid
set key outside
#
set title 'Comparison : No of Threads v Time taken'
set ylabel 'Time taken / s'
set xlabel 'no of Threads'
set origin 0,0
set key outside
plot 'logDir/pdwan-20140714.052102-data-A3-omp-1D.dat' u 2:3 t 'simple' w l
lw 0.5 lc rgb 'blue',  'logDir/pdwan-20140714.052102-data-A3-omp-1D.dat' u
2:5 t 'dgemm' w l lw 0.5 lc rgb 'red'
#
pause -1
```

Plotting the folliowing we get graphs for matrix size v time and also for no of threds v time (using -i so that each iteration uses the same values for A[i] and B[i] where I is the value of the column.

Matrix Size	no of Threads	Time / manual	Infinity Norm / manual	Time /dgemm	Infinity Norm / dgemm
10	2	0.000245	3025	0.000549	3025
10	2	0.002002	3025	0.002351	3025
10	2	0.001001	3025	0.001316	3025
20	2	0.001562	44100	0.002868	44100
20	2	0.000634	44100	0.001808	44100
20	2	0.000673	44100	0.001917	44100
30	10	0.000718	216225	0.002673	216225
30	10	0.000804	216225	0.003045	216225
30	10	0.000774	216225	0.003035	216225
40	10	0.001076	672400	0.004412	672400
40	10	0.001097	672400	0.004799	672400
40	10	0.001169	672400	0.004890	672400
50	10	0.001539	1.62562e+06	0.007291	1.62562e+06
50	10	0.001658	1.62562e+06	0.007325	1.62562e+06
50	10	0.001525	1.62562e+06	0.007136	1.62562e+06

Thankfully for Linux (Ubuntu) – I could install and run GNUplot locally.

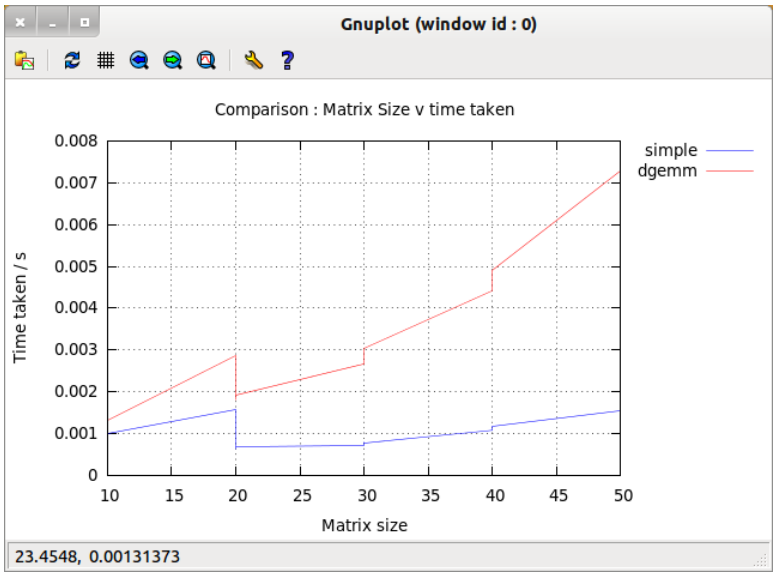
Screen shots of each were taken and added to the sections GNUplot graphs.

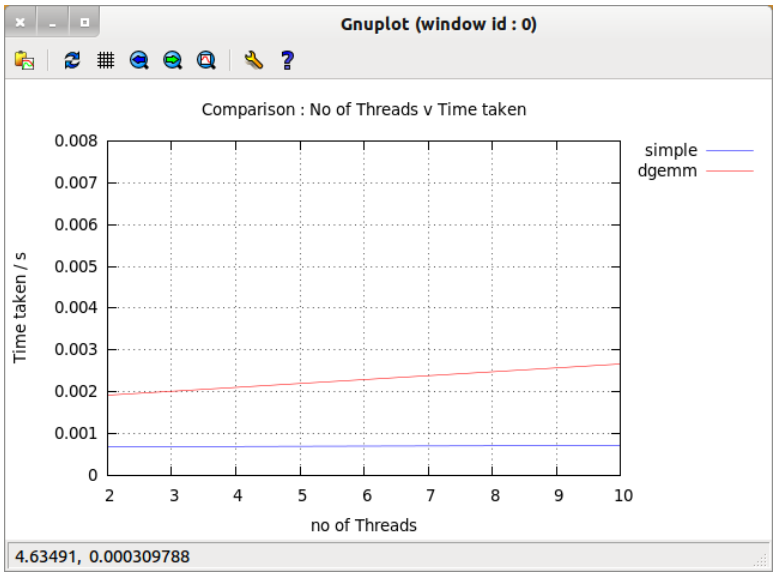
Summary Results :

Build/plot:	• The dependence of the execution time of the program on the matrix size n. • The speedup over a serial counterpart of the program.
Variant :	• *One-step algorithm. No intermediate resulting matrix.* • *Left matrix is horizontally partitioned* • *The maximum absolute row sum norm (aka infinity-norm):* $\|\|A\|\|_\infty = \max_{0 \le i < n} \sum_{j=0}^{n-1} \|a_{ij}\|$
Infinity norm	*Sum the absolute values along each row and then take the biggest answer.* *Example:* *A =* $\begin{vmatrix} 1 & -7 \\ -2 & -3 \end{vmatrix}$ *then* *matrix norm of A = max (1 + |-7|, |-2| + |-3|) = max (8, 5) =* ***8***

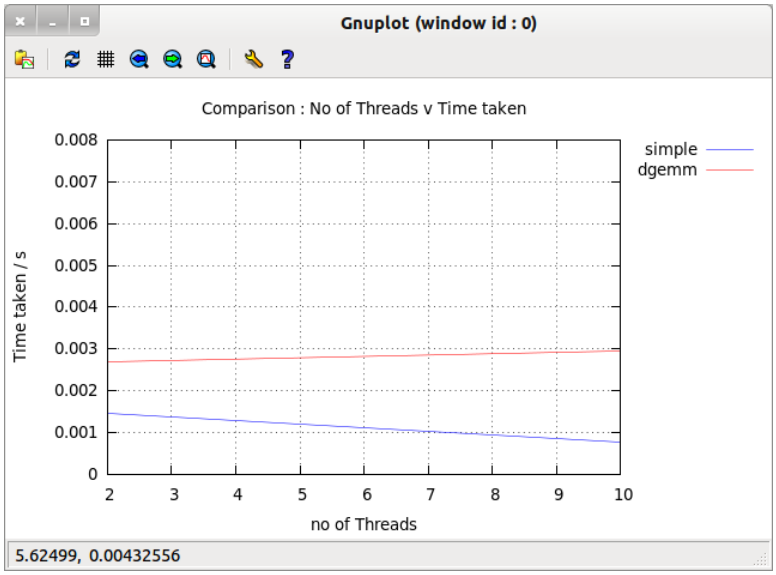
GNUplot Graphs

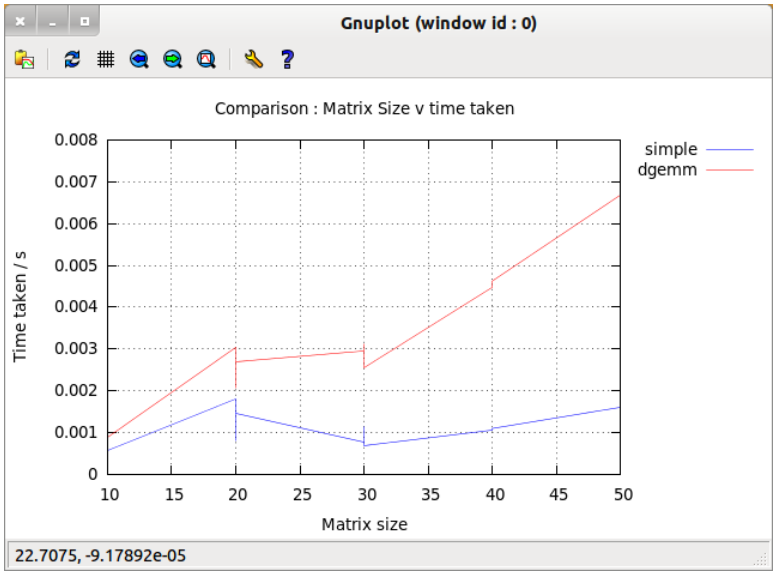
Graph for logDir/pdwan-20140714.051601-data-A3-omp-1D.dat





Using different data set, the following is obtained :





Conclusions

The increase in size of the matrix is after a while no longer negates the increase in time taken. This is evident when the matrix size is increased to 1,000 x 1,000. Also the value was alittle too large to graph. The following is an example of this.

Finally, there is only a finite number of threads available in any one system.

Matrix Size	no of Threads	Time / manual	Infinity Norm / manual	Time /dgemm	Infinity Norm / dgemm
50	10	0.001546	1.62562e+06	0.007661	1.62562e+06
50	10	0.001771	1.62562e+06	0.009032	1.62562e+06
50	10	0.001597	1.62562e+06	0.007557	1.62562e+06
100	10	0.007191	2.55025e+07	0.030856	2.55025e+07
100	10	0.010156	2.55025e+07	0.035289	2.55025e+07
100	10	0.008451	2.55025e+07	0.039047	2.55025e+07
500	20	1.363494	1.56876e+10	2.859575	1.56876e+10
500	20	1.370261	1.56876e+10	2.927285	1.56876e+10
500	20	1.325166	1.56876e+10	3.256134	1.56876e+10

Matrix Size	no of Threads	Time / manual	Infinity Norm / manual	Time /dgemm	Infinity Norm / dgemm
1000	20	12.946991	2.505e+11	21.963079	2.505e+11
1000	20	12.964027	2.505e+11	22.225876	2.505e+11
1000	20	12.597507	2.505e+11	22.171022	2.505e+11

Appendix I – Validate Results

Spot check only using 6x6 matrices, initializing matrices |A| and |B| using successive column values.

Build using :

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help

[pdwan@csserver Assignment3]$  gcc -I/home/cs/khasanov/libs/CBLAS/src A3-omp-1D.c -o A3-omp-1D  /home/cs/khasanov/libs
/cblas_LINUX.a  /usr/lib/libblas.a -lgfortran -fopenmp
[pdwan@csserver Assignment3]$
[pdwan@csserver Assignment3]$ ./A3-omp-1D -i 6 3  m0713.3.txt d0713.3.dat


# RUNNING :     ./A3-omp-1D -i 6 3
# CREATE MATRICES ...
# INITIALIZE MATRICES ...
# RESULTS : complex manual calculation ...
# RESULTS : BLAS/ATLAS calculation -
#               |Matrix|  |Threads|  Time/manual  Inf Norm/manual  Time/dgemm  Inf Norm/dgemm
# Results:      6       3       0.001521s       441     0.001662s       441
# CLEAN-UP ...
[pdwan@csserver Assignment3]$
```

Resulting summary timing data file contains :

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help
#  ----------------------------------------------------------------------------------------------------
#
# Program :      A3-omp-1D
# where :        .dat contains timing data & .txt contains matrix values
#  ----------------------------------------------------------------------------------------------------
# |Matrix|       |Threads|        Time/manual      Inf Norm/manual          Time/dgemm       Inf Norm/dgemm
#
6       3       0.001521s       441     0.001662s       441
~
                                                                                          1,1           All
```

Validating results gives :

Source matrix initialized to value of row for each column

A	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6				
1	1	2	3	4	5	6				
2	1	2	3	4	5	6				
3	1	2	3	4	5	6				
4	1	2	3	4	5	6				
5	1	2	3	4	5	6				
6										
7										
8										
9										

Program – calculate results using dot product

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	21	42	63	84	105	126	0	0	0	0
1	21	42	63	84	105	126	0	0	0	0
2	21	42	63	84	105	126	0	0	0	0
3	21	42	63	84	105	126	0	0	0	0
4	21	42	63	84	105	126	0	0	0	0
5	21	42	63	84	105	126	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

MANUAL

Program – calculate results using for loops

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	21	42	63	84	105	126				
1	21	42	63	84	105	126				
2	21	42	63	84	105	126				
3	21	42	63	84	105	126				
4	21	42	63	84	105	126				
5	21	42	63	84	105	126				
6										
7										
8										
9										

CBLAS

Program – calculate results using clas

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	21	42	63	84	105	126				
1	21	42	63	84	105	126				
2	21	42	63	84	105	126				
3	21	42	63	84	105	126				
4	21	42	63	84	105	126				
5	21	42	63	84	105	126				
6										
7										
8										
9										

Source matrix initialized to value of row for each column

B	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6				
1	1	2	3	4	5	6				
2	1	2	3	4	5	6				
3	1	2	3	4	5	6				
4	1	2	3	4	5	6				
5	1	2	3	4	5	6				
6										
7										
8										
9										

Infinity Norm : max of |total of each row|

MANUAL	CBLAS
441	441
441	441
441	441
441	441
441	441
441	441
0	0
0	0
0	0
0	0
441	441

Program – difference from dot.product

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Program – difference from dot-product

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Running again with -r (random number generation from 1 to 10) :

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help

# RUNNING :     ./A3-omp-1D -r 6 3
# CREATE MATRICES ...
# INITIALIZE MATRICES ...
# RESULTS : complex manual calculation ...
# RESULTS : BLAS/ATLAS calculation -
#               |Matrix|  |Threads|  Time/manual  Inf Norm/manual  Time/dgemm  Inf Norm/dgemm
# Results:      6       3       0.009592s       1396    0.010703s       1396
# CLEAN-UP ...
[pdwan@csserver Assignment3]$ clear
```

Note that the timing calculations are added to the .dat file *(as the same filenames are used)*:

```
pdwan@csserver:~/exercises/Assignment3
File  Edit  View  Search  Terminal  Help
# ------------------------------------------------------------------------------------------------------
#
# Program :      A3-omp-1D
# where :        .dat contains timing data & .txt contains matrix values
# ------------------------------------------------------------------------------------------------------
# |Matrix|       |Threads|        Time/manual      Inf Norm/manual          Time/dgemm       Inf Norm/dgemm
#
6        3        0.001521s        441      0.001662s        441
6        3        0.001246s        1396     0.001480s        1396
                                                                                    1,1            Top
```

Again, validation gives :

Source matrix initialized to value of row for each column

A	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	6	7	8	9	10
4	1	2	3	4	5	6	7	8	9	10
5	1	2	3	4	5	6	7	8	9	10
6	1	2	3	4	5	6	7	8	9	10
7	1	2	3	4	5	6	7	8	9	10
8	1	2	3	4	5	6	7	8	9	10
9	1	2	3	4	5	6	7	8	9	10

Program – calculate results using dot product

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	55	110	165	220	275	330	385	440	495	550
1	55	110	165	220	275	330	385	440	495	550
2	55	110	165	220	275	330	385	440	495	550
3	55	110	165	220	275	330	385	440	495	550
4	55	110	165	220	275	330	385	440	495	550
5	55	110	165	220	275	330	385	440	495	550
6	55	110	165	220	275	330	385	440	495	550
7	55	110	165	220	275	330	385	440	495	550
8	55	110	165	220	275	330	385	440	495	550
9	55	110	165	220	275	330	385	440	495	550

MANUAL

Program – calculate results using for loops

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	55	110	165	220	275	330	385	440	495	550
1	55	110	165	220	275	330	385	440	495	550
2	55	110	165	220	275	330	385	440	495	550
3	55	110	165	220	275	330	385	440	495	550
4	55	110	165	220	275	330	385	440	495	550
5	55	110	165	220	275	330	385	440	495	550
6	55	110	165	220	275	330	385	440	495	550
7	55	110	165	220	275	330	385	440	495	550
8	55	110	165	220	275	330	385	440	495	550
9	55	110	165	220	275	330	385	440	495	550

CBLAS

Program – calculate results using clas

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	55	110	165	220	275	330	385	440	495	550
1	55	110	165	220	275	330	385	440	495	550
2	55	110	165	220	275	330	385	440	495	550
3	55	110	165	220	275	330	385	440	495	550
4	55	110	165	220	275	330	385	440	495	550
5	55	110	165	220	275	330	385	440	495	550
6	55	110	165	220	275	330	385	440	495	550
7	55	110	165	220	275	330	385	440	495	550
8	55	110	165	220	275	330	385	440	495	550
9	55	110	165	220	275	330	385	440	495	550

Source matrix initialized to value of row for each column

B	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	6	7	8	9	10
4	1	2	3	4	5	6	7	8	9	10
5	1	2	3	4	5	6	7	8	9	10
6	1	2	3	4	5	6	7	8	9	10
7	1	2	3	4	5	6	7	8	9	10
8	1	2	3	4	5	6	7	8	9	10
9	1	2	3	4	5	6	7	8	9	10

Infinity Norm : max of |total of each row|

MANUAL	CBLAS
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025
3025	3025

Program – difference from dot product

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Program – difference from dot-product

C_{ijk}	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

APPENDIX II – ACKNOWLEDGEMENTS

- wikipedia.org
- Geln McLachlan – gnuplot tutorials (youtube)
- Stackoverflow.com – examples of Open MP