

# **COMP-40730 HPC**

## **REPORT FOR ASSIGNMENT 2**

Author: Paula Dwan  
Due date: 09-July-2014  
Lecturer: Alexey Lastovetsky  
Subject: COMP-40730 High Performance Computing  
College: University College Dublin

## CONTENTS

<b>Contents.....</b>	<b>2</b>
<b>Exercise.....</b>	<b>3</b>
<b>Overview of Computations Obtained and How.....</b>	<b>4</b>
Assignment Execution.....	5
Data Results Obtained.....	5
Results Evaluated.....	5
GNUplot Execution.....	6
<b>Summary Results :.....</b>	<b>8</b>
GNUplot Graphs.....	8
Conclusions.....	8
<b>Appendix I – Validate Results.....</b>	<b>9</b>

## EXERCISE

Write a parallel Pthreads program computing the norm of the product of two  $n \times n$  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):

$$\|A\|_1 = \max_{0 \leq j < n} \sum_{i=0}^{n-1} |a_{ij}|$$

- (b) The maximum absolute row sum norm (aka infinity-norm):

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

## OVERVIEW OF COMPUTATIONS OBTAINED AND HOW

Assignment 2 basically involved (for me) writing one program which utilized pthreads when calculating manually and BLAS when calculating otherwise. :

### A2-pthreads.-1D.c

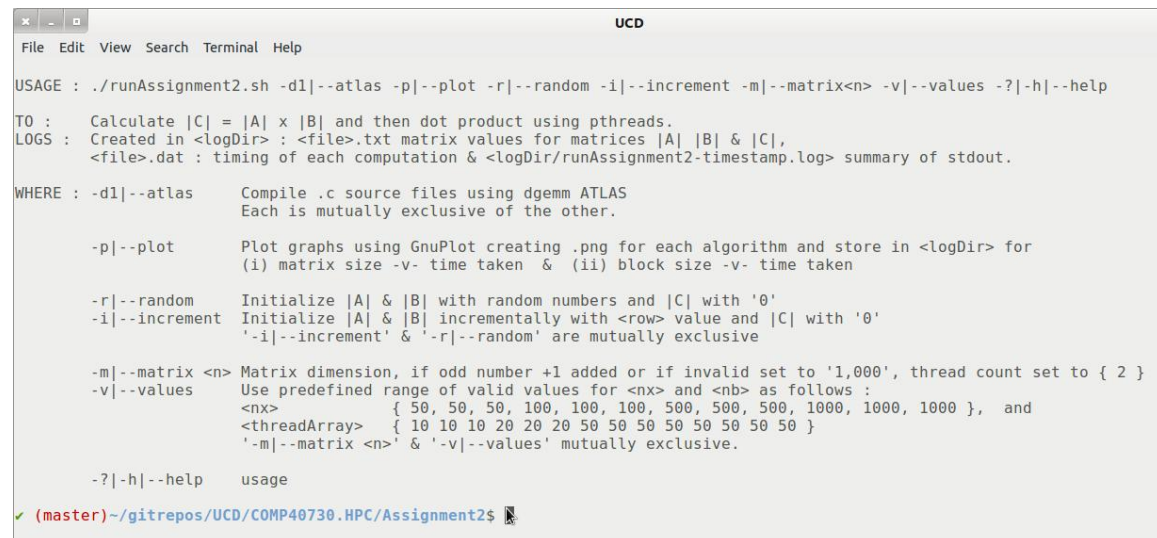
<b>Algorithm</b> <i>using additional variable</i>	<pre>for (i=0; i&lt;rows; i++) {     for (j=0; j&lt;cols; j++)     {         double sum = 0.0;         for (k=0; k&lt;rows; k++)         {             sum+= (A [i][k]) * (B [k][j]);         }         C [i][j] = sum;     } }</pre>
--	---

|C| matrix was calculated using the algorithm as covered in the first assignment. The same matrix computation was implemented using cblas/atlas. Time taken to calculate |C| which is done using pthreads. Calculating the maximum of each value (absolute) in each row was done simultaneously. The time taken to complete was recorded for each manual and dgemm iteration and plotted in a graph.

Thread count is set to a range of values where a series of matrix sizes is used via the script, while if only one matrix size value was entered via the script then the number of threads was set to 2. Also the matrix size if not even is incremented in value by one.

## ASSIGNMENT EXECUTION

Each program was executed multiple times using the script `./runAssignment2.sh`. This has multiple options and the syntax and usage follows:



```
UCD
File Edit View Search Terminal Help

USAGE : ./runAssignment2.sh -d1|--atlas -p|--plot -r|--random -i|--increment -m|--matrix<n> -v|--values -?|-h|--help

TO : Calculate |C| = |A| x |B| and then dot product using pthreads.
LOGS : Created in <logDir> : <file>.txt matrix values for matrices |A| |B| & |C|,
      <file>.dat : timing of each computation & <logDir>/runAssignment2-timestamp.log summary of stdout.

WHERE : -d1|--atlas      Compile .c source files using dgemm ATLAS
      Each is mutually exclusive of the other.

      -p|--plot          Plot graphs using GnuPlot creating .png for each algorithm and store in <logDir> for
      (i) matrix size -v- time taken & (ii) block size -v- time taken

      -r|--random        Initialize |A| & |B| with random numbers and |C| with '0'
      -i|--increment      Initialize |A| & |B| incrementally with <row> 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 '1,000', thread count set to { 2 }
      -v|--values         Use predefined range of valid values for <nx> and <nb> as follows :
      <nx>                { 50, 50, 50, 100, 100, 100, 500, 500, 500, 500, 1000, 1000 }, and
      <threadArray>       { 10 10 10 20 20 20 50 50 50 50 50 50 }
      '-m|--matrix <n>' & '-v|--values' mutually exclusive.

      -?|-h|--help       usage

✓ (master)~/gitrepos/UCD/COMP40730.HPC/Assignment2$
```

Execute this script in the home directory of Assignment 2, retaining the overall directory structure when unzipping.

## DATA RESULTS 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 sub-directory /Results. File naming convention is :

<data log file name>      pdwan-<time>-values-<A2-pthreads-1D->-<iteration>.txt

example:                      pdwan-20140708.015835-values-A2-pthreads-1D-8.txt

Note that if predefined range of valid values is not used then there is only one iteration. Single iteration also applies where the user enters arbitrary, valid values for matrix size and block size for each individual .c program and does not use the scripts. (*This was mainly used to validate the results using a 10x10 matrix.*)

## RESULTS EVALUATED

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-<A2-pthreads-1D->-<iteration>.dat

example:                      pdwan-20140708.015835-timing-A2-pthreads-1D-8.dat

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 :

<b>Sample .dat file</b>	# Program running :     A2-pthreads.c			
	# File name is:         pdwan-A2-pthreads-20140628-022426-timing-0.dat			
	#			
	# Computation of infinity norm of mmatrix C using pthreads			
	#			
	# matrix size	block size	Time/manual	Time / dgemm
	50	2		
	50	5		
	50	10		
	100	5		
	100	10		
	100	20		
	500	10		
	500	20		
	500	50		
	1000	10		
	1000	50		
	1000	100		

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.

<b>Sample GNUplot program execution</b>	# To execute, launch GNUpopt and run :			
	# gnuplot> load <filename.gp>			
	# making sure that the data file name used is updated if needed.			
	unset log			
	unset label			
	set xtic auto			
	set ytic auto			
	set grid			
	set title "Comparison of time taken for manual and dgemm computation \n			
	for matrix size and block size using pthreads"			
	set xlabel "Time taken / ms"			
	set ylabel "size of block / matrix"			
	plot			
	'pdwan-A2-pthreads-20140628-022426-timing-0.dat'			
	u 1:3 t 'Matrix : manual' w l lw 0.5 lc rgb 'blue'			
	'pdwan-A2-pthreads-20140628-022426-timing-0.dat'			
	u 2:3 t 'Block: manual' w l lw 0.5 lc rgb 'green'			
	'pdwan-A2-pthreads-20140628-022426-timing-0.dat'			
	u 1:4 t 'Matrix : dgemm' w l lw 0.5 lc rgb 'black'			
	'pdwan-A2-pthreads-20140628-022426-timing-0.dat'			
	u 2:4 t 'Block: manual' w l lw 0.5 lc rgb 'red'			

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 :

<b>Build/plot:</b>	<ul style="list-style-type: none"><li>• The dependence of the execution time of the program on the matrix size n.</li><li>• The speedup over a serial counterpart of the program.</li></ul>
<b>Variant :</b>	<ul style="list-style-type: none"><li>• <i>One-step algorithm. No intermediate resulting matrix.</i></li><li>• <i>Left matrix is horizontally partitioned</i></li><li>• <i>The maximum absolute row sum norm (aka infinity-norm):</i><math display="block">\ A\ _{\infty} = \max_{0 \leq i &lt; n} \sum_{j=0}^{n-1}  a_{ij} </math></li></ul>
<b>Infinity norm</b>	<p><i>Sum the absolute values along each row and then take the largest value as the answer.</i></p> <p><i>Example:</i>      <math>A = \begin{vmatrix} 1 &amp; -7 \\ -2 &amp; -3 \end{vmatrix}</math></p> <p><i>then matrix norm of A = <math>\max ( 1 +  -7 ,  -2  +  -3  ) = \max ( 8, 5 ) = \underline{\underline{8}}</math></i></p>

## GNU PLOT GRAPHS

## CONCLUSIONS

## **APPENDIX I – VALIDATE RESULTS**

Spot check only using 10x10 matrix for random number generation to initialize matrices |A| and |B| and also row values for each.