COMP-40730 HPC

Report for Assignment 4

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

Contents

EXERCISE

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

- p processors are involved in the computations.
- The 1-dimensional parallel algorithm of matrix multiplication is employed:
    - the matrices are identically and equally partitioned in one dimension into p horizontal slices
    - there is one-to-one mapping between the partitions and the processors
    - each processor is responsible for computation of the corresponding slice of the resulting matrix

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):

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

    - (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 4 basically involved (for me) writing one program, which contained the code for manual straight-forward IJK computation, calculation of matrix |C| using dgemm and calculation of matrix |C| using MPI. I then compared the time taken to calculate matrix |C| for each method using [N]x[N] matrices |A| and |B| of different sizes. :

1. **manual straight-forward IJK computation**

   Implementation of a straight forward matrix nxn multiplication :

   Code

```
for (ni=0 ; ni<rows ; ni++)
{
   for (nj=0 ; nj<cols ; nj++)
   {
      double sum = 0.0 ;
      for (nk=0 ; nk<rows ; nk++)
      {
         sum+= (A[(ni*rows)+nk]) * (B[(nk*rows)+nj]) ;
      }
      C[(ni*rows)+nj] = sum ;
   }
}
```

2. **Dgemm for straight-forward IJK computation**

   I used dgemm compiled for atlas or for cblas.

   Code

```
   int ni, nj ;
// m, n, k :    local integers indicating the size of the matrices for
// rows x columns :: A :  m x  k, B :  k x n, C:  m x n
// Here, m = n = k = rows = columns = <nx> = <ny> as supplied
   int lm = rows, ln = rows ;
// la_offset, lb_offset, lc_offset :
// Leading dimension of matrix A, B or C respectively, or the number of elements
// between successive rows for row-major storage or columns for column-major
// storage.
   int la_offset = rows, lb_offset = cols, lc_offset = rows ;
   int ALPHA=1.0 ;
   int BETA=0.0 ;

   cblas_dgemm( CblasRowMajor, CblasNoTrans, CblasNoTrans, lm, ln, ln, ALPHA, \
      A, la_offset, B, lb_offset, BETA, C, lc_offset) ;
```

3. **MPI calculation**

   This was completed using a data structure, creating a new thread to correspond to the each slice.

   Code

```
MPI_Barrier ()
MPI_Bcast ()
MPI_Comm_rank ()
MPI_Comm_size ()
MPI_Finalize ()
MPI_Gather ()
MPI_Init ()
MPI_Scatter ()
MPI_Wtime()
```

   Note: *Please see section for detailed description of code used : Overview of MPI functions used.*

For options 2. and 3., the matrix |C| was calculated (cblas by default otherwise the use could decide to re-build and execute using cblas or atlas).

Thus results were obtained for each option 1., 2. and 3. using the same source |A| and |B| to ensure like was compared to like.

Multiple implementation of each .c program was enabled using ***./runAssignment4.sh,*** for example :

```
$ ./runAssignment4.sh -i -v
```

This runs the c. program using cblas for incremental column values, using predefined settings for matrix for each implementation.

Single implementation is completed using **A4-mpi-1D.c.**

```
$ ./A4-mpi-1D-cblas -i 10 file.txt file.dat
```

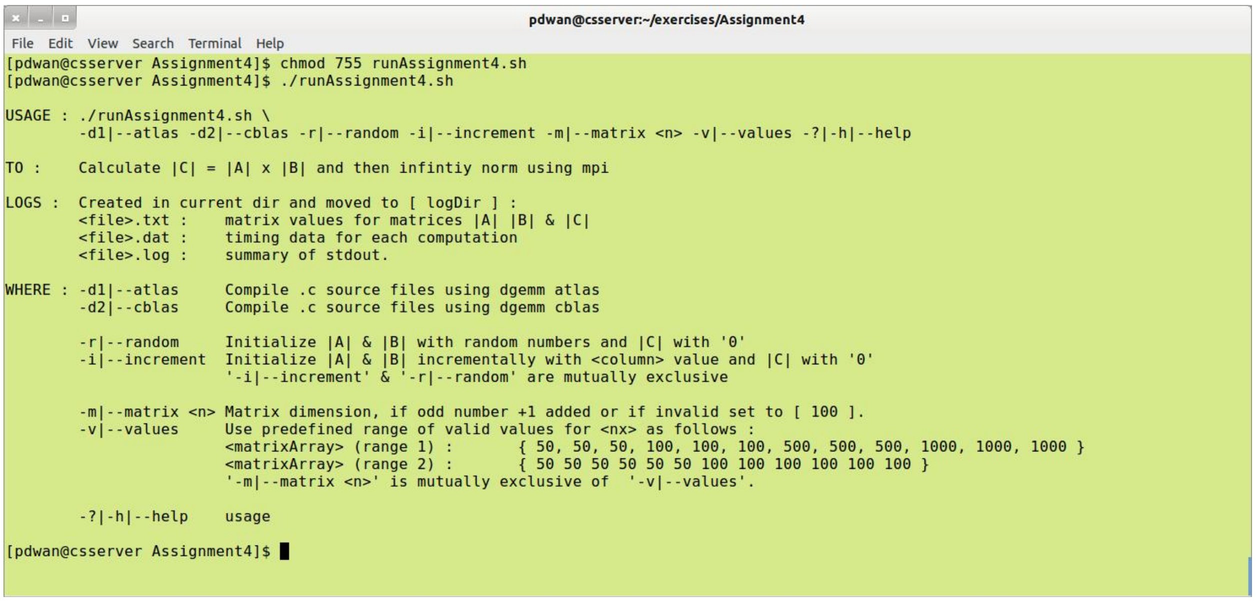
This is compiled using ***mpicc*** for atlas and cblas, as follows :

```
mpicc -I/home/cs/khasanov/libs/CBLAS/src A4-mpi-1D.c -o A4-mpi-1D-cblas        \
     /home/cs/khasanov/libs/cblas_LINUX.a  /usr/lib/libblas.a -lgfortran
mpicc -o A4-mpi-1D-atlas A4-mpi-1D.c -I/home/cs/khasanov/libs/ATLAS/include/        \
     -L/home/cs/khasanov/libs/ATLAS/lib/Linux_UNKNOWNSSE2_4/ -lcblas -latlas -lm -O3
```

Assignment Execution

The compiled .c program ./A4-mpi-1D was executed multiple times standalone or using the script *./runAssignment4.sh* to obtain as wide a range of time taken to calculate |C| using each algorithm.

This has multiple options and the syntax and usage follows :

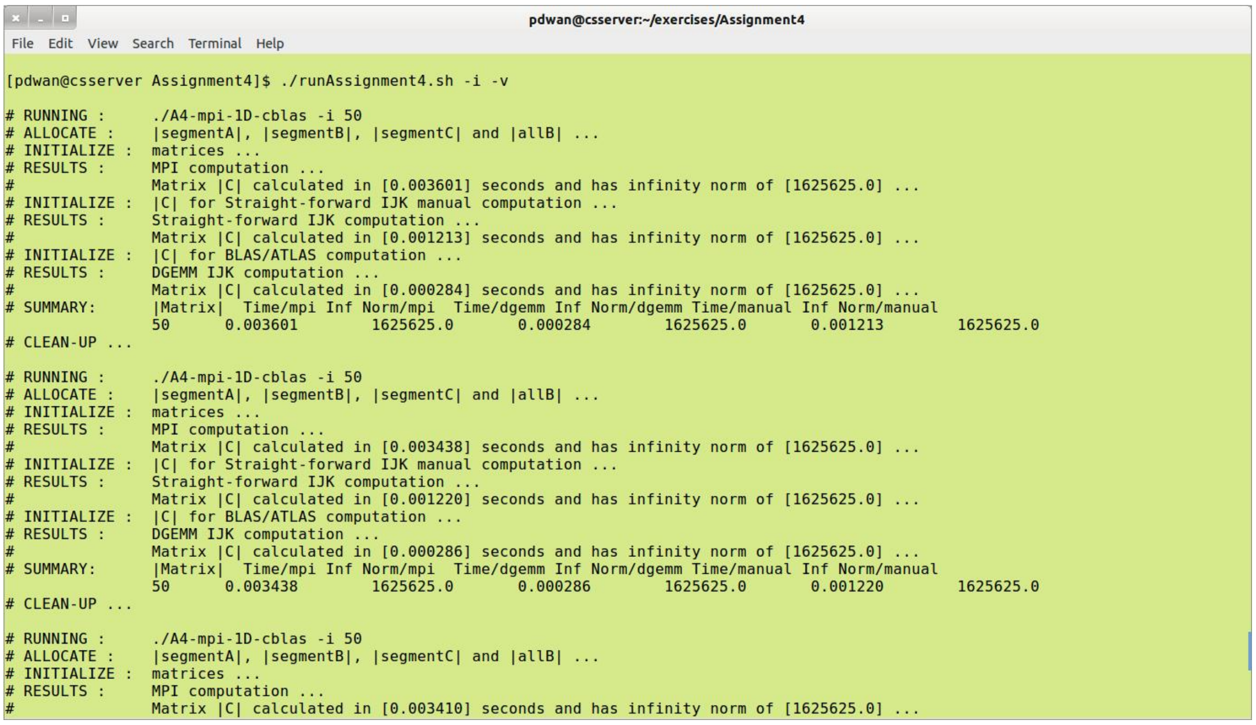

pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```
[pdwan@csserver Assignment4]$ chmod 755 runAssignment4.sh
[pdwan@csserver Assignment4]$ ./runAssignment4.sh

USAGE : ./runAssignment4.sh \
        -d1|--atlas -d2|--cblas -r|--random -i|--increment -m|--matrix <n> -v|--values -?|-h|--help

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

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 : -d1|--atlas     Compile .c source files using dgemm atlas
        -d2|--cblas     Compile .c source files using dgemm cblas

        -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 [ 100 ].
        -v|--values     Use predefined range of valid values for <nx> as follows :
                        <matrixArray> (range 1) :       { 50, 50, 50, 100, 100, 100, 500, 500, 500, 1000, 1000, 1000 }
                        <matrixArray> (range 2) :       { 50 50 50 50 50 50 100 100 100 100 100 100 }
                        '-m|--matrix <n>' is mutually exclusive of  '-v|--values'.

        -?|-h|--help    usage

[pdwan@csserver Assignment4]$ █
```

Execute this script in the home directory of Assignment 4.

Sample execution follows for :

`$ ./runAssignment4.sh -i -v`


pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```
[pdwan@csserver Assignment4]$ ./runAssignment4.sh -i -v

# RUNNING :     ./A4-mpi-1D-cblas -i 50
# ALLOCATE :    |segmentA|, |segmentB|, |segmentC| and |allB| ...
# INITIALIZE :  matrices ...
# RESULTS :     MPI computation ...
#               Matrix |C| calculated in [0.003601] seconds and has infinity norm of [1625625.0] ...
# INITIALIZE :  |C| for Straight-forward IJK manual computation ...
# RESULTS :     Straight-forward IJK computation ...
#               Matrix |C| calculated in [0.001213] seconds and has infinity norm of [1625625.0] ...
# INITIALIZE :  |C| for BLAS/ATLAS computation ...
# RESULTS :     DGEMM IJK computation ...
#               Matrix |C| calculated in [0.000284] seconds and has infinity norm of [1625625.0] ...
# SUMMARY:      |Matrix|  Time/mpi Inf Norm/mpi  Time/dgemm Inf Norm/dgemm Time/manual Inf Norm/manual
                50      0.003601        1625625.0       0.000284        1625625.0       0.001213        1625625.0
# CLEAN-UP ...

# RUNNING :     ./A4-mpi-1D-cblas -i 50
# ALLOCATE :    |segmentA|, |segmentB|, |segmentC| and |allB| ...
# INITIALIZE :  matrices ...
# RESULTS :     MPI computation ...
#               Matrix |C| calculated in [0.003438] seconds and has infinity norm of [1625625.0] ...
# INITIALIZE :  |C| for Straight-forward IJK manual computation ...
# RESULTS :     Straight-forward IJK computation ...
#               Matrix |C| calculated in [0.001220] seconds and has infinity norm of [1625625.0] ...
# INITIALIZE :  |C| for BLAS/ATLAS computation ...
# RESULTS :     DGEMM IJK computation ...
#               Matrix |C| calculated in [0.000286] seconds and has infinity norm of [1625625.0] ...
# SUMMARY:      |Matrix|  Time/mpi Inf Norm/mpi  Time/dgemm Inf Norm/dgemm Time/manual Inf Norm/manual
                50      0.003438        1625625.0       0.000286        1625625.0       0.001220        1625625.0
# CLEAN-UP ...

# RUNNING :     ./A4-mpi-1D-cblas -i 50
# ALLOCATE :    |segmentA|, |segmentB|, |segmentC| and |allB| ...
# INITIALIZE :  matrices ...
# RESULTS :     MPI computation ...
#               Matrix |C| calculated in [0.003410] seconds and has infinity norm of [1625625.0] ...
```

Please retain the overall directory structure when unzipping.

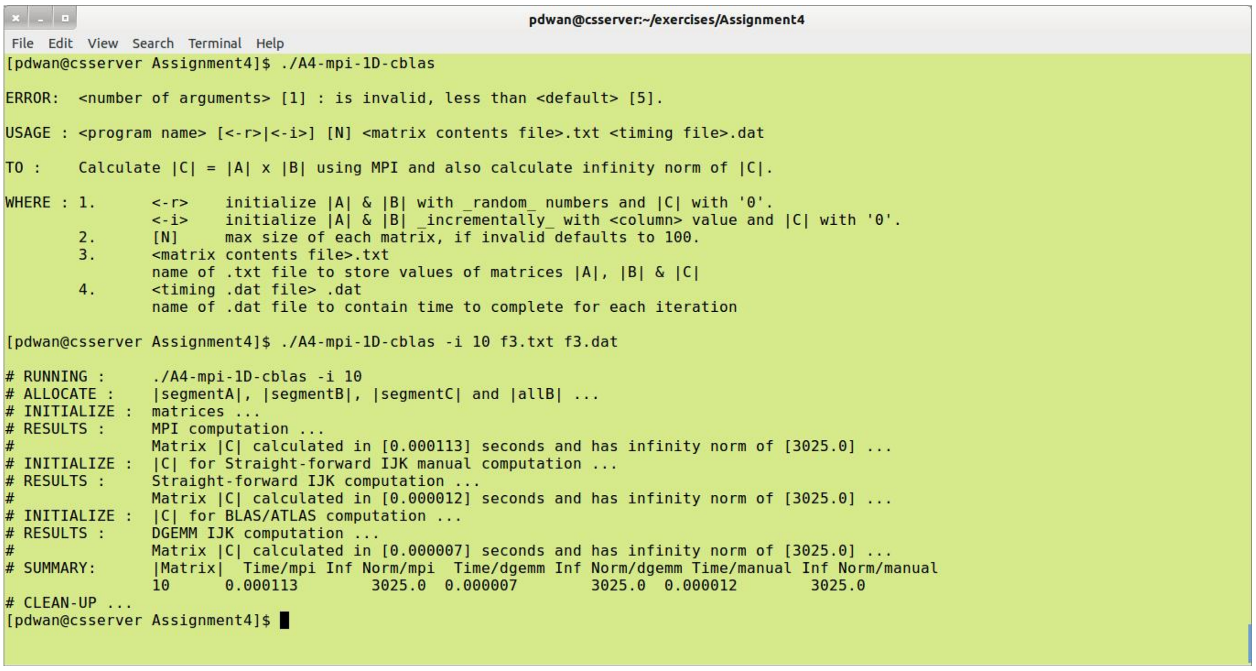
Note that the script `./runAssignment4.sh` allows two types of implementation

- Multiple iteration : use the switch <-v|--values>, when a predefined range applies for [N] : matrix size.

- Single iteration : use the switch <-m|--matrix> [N] where the user specifies value for [N] : matrix size.

Running A4-mpi-1D-<atlas | cblas> : standalone

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


pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```
[pdwan@csserver Assignment4]$ ./A4-mpi-1D-cblas

ERROR:  <number of arguments> [1] : is invalid, less than <default> [5].

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

TO :    Calculate |C| = |A| x |B| using MPI 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 100.
        3.      <matrix contents file>.txt
                name of .txt file to store values of matrices |A|, |B| & |C|
        4.      <timing .dat file> .dat
                name of .dat file to contain time to complete for each iteration

[pdwan@csserver Assignment4]$ ./A4-mpi-1D-cblas -i 10 f3.txt f3.dat

# RUNNING :     ./A4-mpi-1D-cblas -i 10
# ALLOCATE :    |segmentA|, |segmentB|, |segmentC| and |allB| ...
# INITIALIZE :  matrices ...
# RESULTS :     MPI computation ...
#               Matrix |C| calculated in [0.000113] seconds and has infinity norm of [3025.0] ...
# INITIALIZE :  |C| for Straight-forward IJK manual computation ...
# RESULTS :     Straight-forward IJK computation ...
#               Matrix |C| calculated in [0.000012] seconds and has infinity norm of [3025.0] ...
# INITIALIZE :  |C| for BLAS/ATLAS computation ...
# RESULTS :     DGEMM IJK computation ...
#               Matrix |C| calculated in [0.000007] seconds and has infinity norm of [3025.0] ...
# SUMMARY:      |Matrix|  Time/mpi Inf Norm/mpi  Time/dgemm Inf Norm/dgemm Time/manual Inf Norm/manual
                10      0.000113        3025.0  0.000007        3025.0  0.000012        3025.0
# CLEAN-UP ...
[pdwan@csserver Assignment4]$ █
```

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>	`Values-<time>-A4-mpi-1D-<iteration>.txt`
example:	`Values-20140715.170928-A4-mpi-1D-0.txt` `f1.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. Each new matrices |A| and |B| and the results in |C| were saved to the data file, thus simple validation using *LibreOffice Calc*.

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

<timing log file name>	`Data-<time>-A4-mpi-1D.dat`

example:	`Data-20140715.171337-A4-mpi-1D.dat`
	`Data-20140715.172124-A4-mpi-1D.dat`
	`f1.dat`

I did not save a separate .dat file for each run of the script for each algorithm. Instead each .dat file contains the time taken for each matrix size for the preset range of values. **./runAssignment4.sh** may be updated with more if needed but the following are those in use at the moment.

```
#   Matrix - range 1
    declare -a NXArray=( 50 50 50 50 50 50 100 100 100 100 100 100 )
#   Matrix size - range 2
    declare -a NXArray=( 50 50 50 100 100 100 500 500 500 500 1000 1000 1000 1000 )
```

For compilation using the script, a suffix of ***-atlas*** indicates compilation for atlas and a suffix of ***-cblas*** indicates that the c program was compiled via cblas.

Finally a log file containing a listing of each algorithm used for that iteration.

After each run, all .log, .txt, .dat and .bup files are copied to the directory *logDir/*.

If ***./A4-mpi-1D-<cblas|atlas>*** is used without the script then .dat and .txt files may be named whatever the user wishes and no .log file applies.

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 if ran standalone.

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

GNUPLOT EXECUTION

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

Sample .dat file	

If wished, the .txt file contains the matrices |A| and |B| used to calculate |C| and the type of computation applicable and the time taken to complete. The .dat file is just a summary of the matrix sizes (when the later is applicable) as well as time taken for each type of computation.

The contents of each .dat was then presented in graphical format using GNUplot, comparing times taken for manual and for BLAS/ATLAS computations.

Sample 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 4
reset
set xtic auto
set ytic auto
set size 1,1
set grid
set key outside
#
set title 'mpi : Matrix size -v- Time taken'
set ylabel 'Time taken / seconds'
set xlabel 'Matrix size'
set xrange [0:1100]
set yrange [0:34]
set xtics (100,200,300,400,500,600,700,800,900,1000,1100)
set origin 0,0
set key outside
plot 'logDir/Data-mpi.dat' u 1:3 t 'manual' w l lw 0.8 lc rgb 'blue',
'logDir/Data-mpi.dat' u 1:5 t 'dgemm' w l lw 0.8 lc rgb 'black',
'logDir/Data-mpi.dat' u 1:7 t 'mpi' w l lw 0.8 lc rgb 'red'
#
pause -1
```

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

Screen shots of each were taken and added to the section Summary Results.

Overview of MPI functions used

Basic Linear Algebra Subprograms (BLAS) are a specified set of low-level subroutines that are used in this exercise for matrix multiplication (Level 3 BLAS). Compilation may be completed using ATLAS (Automatically Tuned Linear Algebra Software) or cBLAS.

So in this exercise, we calculate :

```
|C| += |A| x |B|
```

Message Passing Interface (MPI) supports communication between processors, i.e.: parallel programming. MPI is a standardized implementation, in this case written in C, and easily applies dgemm to calculate the portion of the matrix A as send to that processor for computation.

In this .c program implementing MPI, we have a fixed number of process [P] using a [N]x[N] sized matrix for |A|, |B|, and |C|. Each processor can communicate via calls to MPI communication primitives. Each we will use collective communication which involves a group of processors.

The MPI functions used are *(not necessarily in order of use)*:

- MPI_Init (&argc, &argv)

  When the program starts only one process is in use (the root process), MPI_Init initializes the run time environment creating the child processes.

  Syntax :

  ```
  int MPI_Init(int *argc, char ***argv)
  ```

  where :

&argc	*argc*	*C/C++ only: Pointer to the number of arguments.* *Number of arguments in program.*
&argv	*argv*	*C/C++ only: Argument vector.* *Listing of arguments applicable to program.*

- MPI_Comm_rank (MPI_COMM_WORLD, &rank)

  Returns the rank of the current process.

  Syntax :

  ```
  int MPI_Comm_rank(MPI_Comm comm, int *rank)
  ```

  where :

MPI_COMM_WORLD	*comm*	*Communicator (handle).* *Default communicator* *communicator of all processes making up the MPI program*
&rank	*rank*	*Rank of the calling process in group of comm (integer).* *Returns the rank of all the processes in the group.*

- MPI_Comm_size (MPI_COMM_WORLD, &size)

  Returns the number of processes requested for the the job

  Syntax :

  ```
  int MPI_Comm_size(MPI_Comm comm, int *size)
  ```

  where :

MPI_COMM_WORLD	*comm*	*Communicator (handle).* *Default communicator* *communicator of all processes making up the MPI program*
&size	*size*	*Number of processes in the group of comm (integer).* *Number of processes in the group*

Ø MPI_Bcast (B, nx*ny, MPI_DOUBLE, 0, MPI_COMM_WORLD)

Broadcasts a message from the root process to all other processes in the group.

Syntax :

```
int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)
```

where :

B	*buffer*	*Starting address of buffer (choice).* *Matrix |B|*
*nx*ny*	*count*	*Number of entries in buffer (integer).* *[N] x [N] matrix size of [row] x [column]*
MPI_DOUBLE	*datatype*	*Data type of buffer (handle).* *Data item is an integer : size of matrix [row] x [column]*
0	*root*	*Rank of broadcast root (integer).* *Root process reference*
MPI_COMM_WORLD	*comm*	*Communicator (handle).* *Default communicator* *communicator of all processes making up the MPI program*

Ø MPI_Scatter (*A, nx*ny/np, MPI_DOUBLE, A + offset, nx*ny/np, MPI_DOUBLE, 0, MPI_COMM_WORLD*)

Sends data from one task to all tasks in a group.

Syntax :

```
int MPI_Scatter(const void *sendbuf, int sendcount, MPI_Datatype sendcount,
    void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

where :

A	*sendbuf*	*Address of send buffer (choice, significant only at root).* *matrix |A| → full*
*nx*ny/np*	*sendcount*	*Number of elements in send buffer (integer, significant only at root).* *Size of buffer – so [row] x [column] / [no. of processors]*
MPI_DOUBLE	*sendtype*	*Datatype of send buffer elements (handle, significant only at root).* *→ here, double.*
A + offset	*recvbuf*	*Address of receive buffer (choice).* *matrix |A| → subset only*
*nx*ny/np*	*recvcount*	*Number of elements in receive buffer (integer).* *size of buffer – so [row] x [column] / [no. of processors]*
MPI_DOUBLE	*recvtype*	*Datatype of receive buffer elements (handle).* *→ here, double.*
0	*root*	*Rank of broadcast root (integer).* *Root process reference*
MPI_COMM_WORLD	*comm*	*Communicator (handle).* *Default communicator* *communicator of all processes making up the MPI program*

Ø MPI_Finalize ()

Closes the run-time environment. No more MPI calls may be made once this function is executed, not even MPI_Init().

Syntax :

```
int MPI_Finalize(void)
```

- Ø MPI_Wtime ()

  MPI_Wtime returns a floating-point number of seconds, representing elapsed wall-clock time since some time in the past.

  Syntax :

  ```
  double MPI_Wtime()
  ```

- Ø MPI_Barrier

  Blocks until all processes in the communicator have reached this routine.

  Syntax :

  ```
  int MPI_Barrier(MPI_Comm comm)
  ```

  where :

MPI_COMM_WORLD	*comm*	*Communicator (handle).* *Default communicator* *communicator of all processes making up the MPI program*

- Ø MPI_Gather (C+offset, nx*ny/np, MPI_DOUBLE, C, nx*ny/np, MPI_DOUBLE, 0, MPI_COMM_WORLD)

  Gather / collect the distributed blocks and return them to the root process.

  Syntax :

  ```
  int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
      void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
  ```

  where :

C + offset	*sendbuf*	*Starting address of send buffer (choice).* *matrix |C| → subset only*
*nx*ny/np*	*sendcount*	*Number of elements in send buffer (integer).* *size of buffer – so [row] x [volumn] / [no. of processors]*
MPI_DOUBLE	*sendtype*	*Datatype of send buffer elements (handle).* *→ here, double.*
C	*recvbuf*	*Address of receive buffer (choice, significant only at root).* *matrix |C| → full*
*nx*ny/np*	*recvcount*	*Number of elements for any single receive (integer, significant only at root).* *As previously,* *size of buffer – so [row] x [volumn] / [no. of processors]*
MPI_DOUBLE	*recvtype r*	*Datatype of recvbuffer elements (handle, significant only at root).* *→ here, double.*
0	*root*	*Rank of receiving process (integer).*
MPI_COMM_WORLD	comm	Communicator (handle). *Default communicator* *communicator of all processes making up the MPI program*

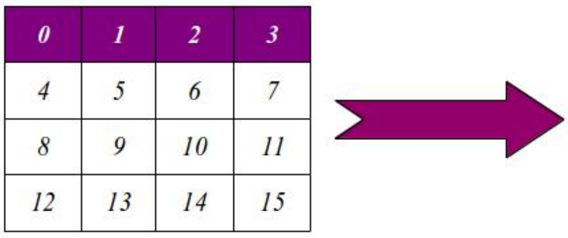
Matrix Sizes Evaluated

When applying the three options I used matrices |A| and |B| of varying sizes from [10x10] to [1000x1000]. The values in each matrix were dependent on the switch

- **-r**
  random from 1 to 10, so each cell regardless of matrix size had a value of 1 to 10 inclusive. This reduced computation time based on large cell values.
- **-i**
  increment based on column index + 1, so all cells in column 1000 had a value of 1001. This could increase computation time of larger matrices as the cell would have comparatively larger values also.

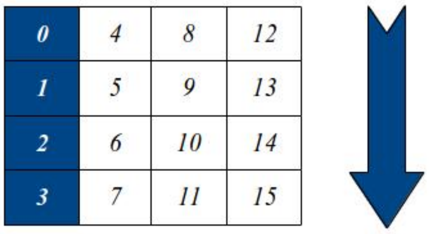
Row-Major as applied

For the straight-forward algorithm of $|\mathbf{C}|_{[ij]} \mathrel{+}= |\mathbf{A}|_{[ik]} * |\mathbf{B}|_{[kj]}$ uses row major as follows (e.g.: [4x4] matrix:



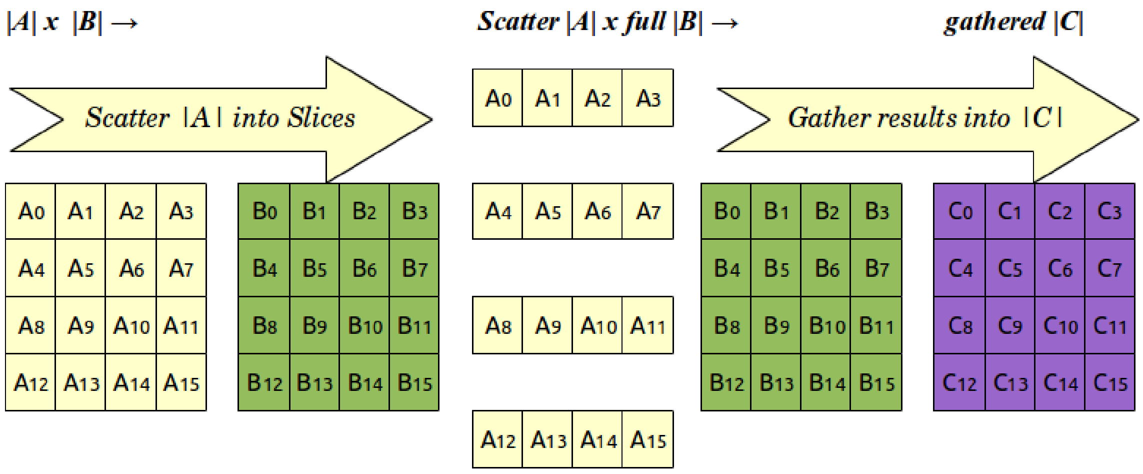
0	*1*	*2*	*3*
4	*5*	*6*	*7*
8	*9*	*10*	*11*
12	*13*	*14*	*15*

Whereas column-major is as follows :



0	*4*	*8*	*12*
1	*5*	*9*	*13*
2	*6*	*10*	*14*
3	*7*	*11*	*15*

So for this assignment (variant applied) we use :



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 largest value as the 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***

Comparison MPI, Open MP & pthreads

As this is the fourth assignment in a series using pthreads, open MP and MPI, it is probably worth comparing each at this stage, at a very high level.

OpenMP implementation utilises thread spawning and thread-to-processor allocation. This means less memory usage and direct thread to thread communication. However, threads must be synchronized and joined, thus increasing the time taken to execute the program particularly for larger matrix sizes. Open MP optimises program performance which runs on a single machine using muptiple CPU's.

PThreads is esentially OpenMP on a lower level. It uses #pragmas (basically, code comments) to indicate to the compiler that this is where threading applies.

Message Passing Interface (MPI), basically divides a complex problem into smaller parts which can be solved on indivdual machines. MPI uses direct inter-process communication via bus messages. Each process (program segement) is standalone and no memory is shared.

GNUplot Graphs – cblas using matrix sizes 50 → 1000

$ ./runAssignment4.sh -i -v

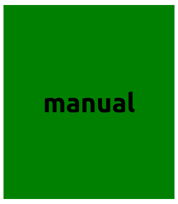
manual

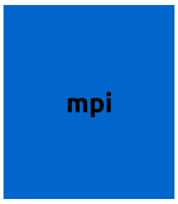
dgemm

mpi

GNUplot Graphs – CBLAS using matrix sizes 50 → 100

$ ./runAssignment4.sh -r -v



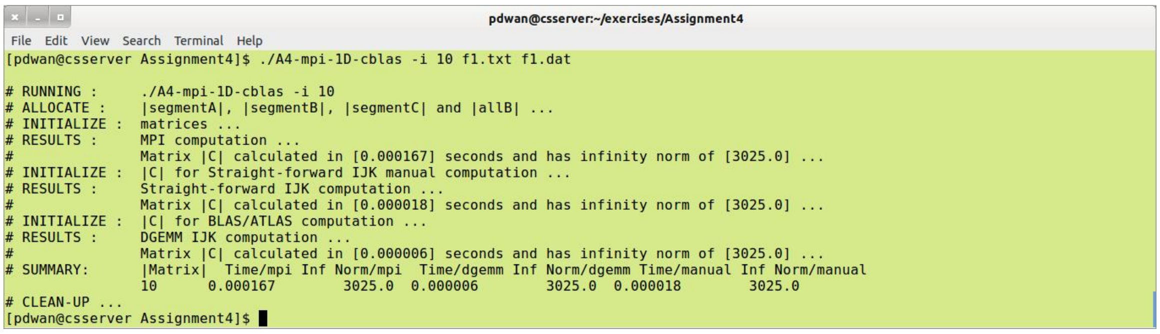


Conclusions

Appendices

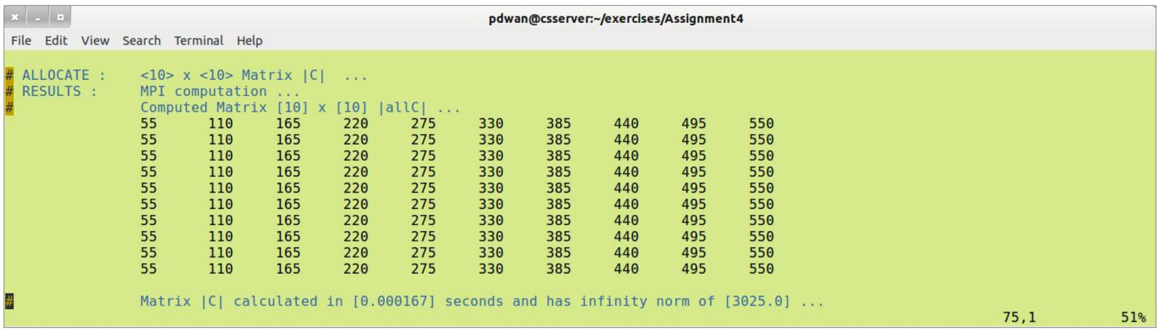
Appendix I – Validate Results

Spot check only using 10x10 matrices, initializing matrices |A| and |B| using successive column values.
Built using :

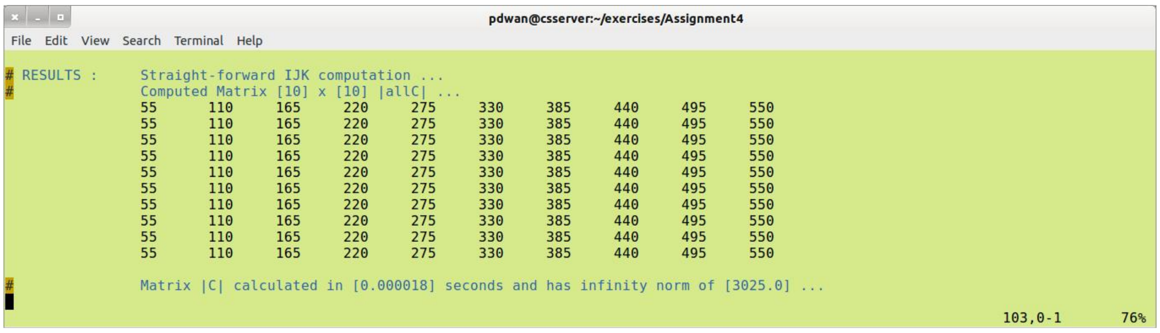

pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```
[pdwan@csserver Assignment4]$ ./A4-mpi-1D-cblas -i 10 f1.txt f1.dat

# RUNNING :     ./A4-mpi-1D-cblas -i 10
# ALLOCATE :    |segmentA|, |segmentB|, |segmentC| and |allB| ...
# INITIALIZE :  matrices ...
# RESULTS :     MPI computation ...
#               Matrix |C| calculated in [0.000167] seconds and has infinity norm of [3025.0] ...
# INITIALIZE :  |C| for Straight-forward IJK manual computation ...
# RESULTS :     Straight-forward IJK computation ...
#               Matrix |C| calculated in [0.000018] seconds and has infinity norm of [3025.0] ...
# INITIALIZE :  |C| for BLAS/ATLAS computation ...
# RESULTS :     DGEMM IJK computation ...
#               Matrix |C| calculated in [0.000006] seconds and has infinity norm of [3025.0] ...
# SUMMARY:      |Matrix|  Time/mpi Inf Norm/mpi  Time/dgemm Inf Norm/dgemm Time/manual Inf Norm/manual
                10      0.000167        3025.0  0.000006        3025.0  0.000018        3025.0
# CLEAN-UP ...
[pdwan@csserver Assignment4]$ 
```

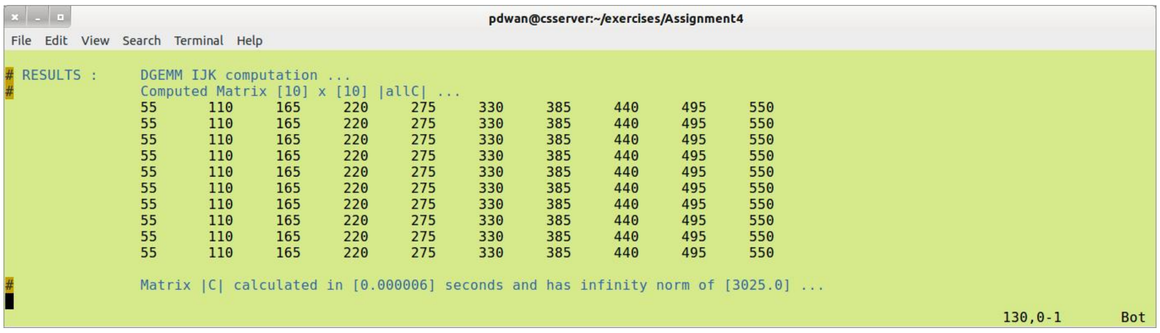
Resulting sample Matrix .txt file contains :


pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```

# ALLOCATE :    <10> x <10> Matrix |C|  ...
# RESULTS :     MPI computation ...
#               Computed Matrix [10] x [10] |allC| ...
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550

#               Matrix |C| calculated in [0.000167] seconds and has infinity norm of [3025.0] ...
                                                                                                             75,1          51%
```

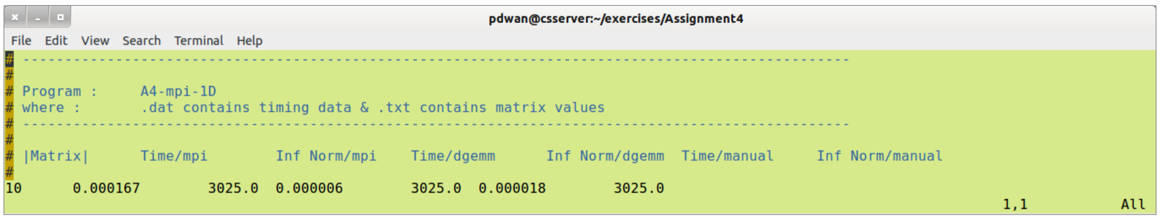

pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```

# RESULTS :     Straight-forward IJK computation ...
#               Computed Matrix [10] x [10] |allC| ...
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550

#               Matrix |C| calculated in [0.000018] seconds and has infinity norm of [3025.0] ...

                                                                                                             103,0-1       76%
```


pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```

# RESULTS :     DGEMM IJK computation ...
#               Computed Matrix [10] x [10] |allC| ...
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550
                55      110     165     220     275     330     385     440     495     550

#               Matrix |C| calculated in [0.000006] seconds and has infinity norm of [3025.0] ...

                                                                                                             130,0-1       Bot
```

Resulting sample summary timing data file contains :


pdwan@csserver:~/exercises/Assignment4
File Edit View Search Terminal Help

```
# ----------------------------------------------------------------------------------------------------
#
# Program :     A4-mpi-1D
# where :       .dat contains timing data & .txt contains matrix values
# ----------------------------------------------------------------------------------------------------
#
# |Matrix|      Time/mpi        Inf Norm/mpi    Time/dgemm      Inf Norm/dgemm  Time/manual     Inf Norm/manual
#
10      0.000167        3025.0  0.000006        3025.0  0.000018        3025.0
                                                                                                             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	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

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	MPI
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025
3,025	3,025	3,025

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

MPI

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

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

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 – Summary of Results Obtained

./A4-mpi-1D-cblas -r 50 → 100

|Matrix|	Time / mpi	Inf Norm / mpi	Time / dgemm	Inf Norm / dgemm	Time / manual	Inf Norm / manual
50	0.003623	89,502.0	0.000283	89,502.0	0.001390	89,502.0
50	0.003733	89,502.0	0.000285	89,502.0	0.001273	89,502.0
50	0.003172	89,502.0	0.000282	89,502.0	0.001216	89,502.0
50	0.003412	89,502.0	0.000290	89,502.0	0.001217	89,502.0
50	0.003678	89,502.0	0.000287	89,502.0	0.001214	89,502.0
50	0.003299	89,502.0	0.000295	89,502.0	0.001214	89,502.0
100	0.018782	353,266.0	0.002029	353,266.0	0.009716	353,266.0
100	0.018608	353,266.0	0.002060	353,266.0	0.009452	353,266.0
100	0.017310	353,266.0	0.002612	353,266.0	0.009707	353,266.0
100	0.018808	353,266.0	0.002283	353,266.0	0.010504	353,266.0
100	0.020819	353,266.0	0.002188	353,266.0	0.010953	353,266.0
100	0.029216	353,266.0	0.002254	353,266.0	0.020780	353,266.0
50	0.003392	89,502.0	0.000284	89,502.0	0.001266	89,502.0
50	0.003450	89,502.0	0.000285	89,502.0	0.001228	89,502.0
50	0.003227	89,502.0	0.000289	89,502.0	0.001234	89,502.0
50	0.003526	89,502.0	0.000286	89,502.0	0.001235	89,502.0
50	0.003154	89,502.0	0.000286	89,502.0	0.001231	89,502.0
50	0.003247	89,502.0	0.000293	89,502.0	0.001258	89,502.0
100	0.017802	353,266.0	0.002062	353,266.0	0.009515	353,266.0
100	0.018403	353,266.0	0.002130	353,266.0	0.009852	353,266.0
100	0.019895	353,266.0	0.002405	353,266.0	0.010686	353,266.0
100	0.019114	353,266.0	0.002475	353,266.0	0.010289	353,266.0
100	0.019182	353,266.0	0.002044	353,266.0	0.009500	353,266.0
100	0.017434	353,266.0	0.002044	353,266.0	0.010240	353,266.0

./A4-mpi-1D-cblas -i 50 → 100

|Matrix|	Time / mpi	Inf Norm / mpi	Time / dgemm	Inf Norm / dgemm	Time / manual	Inf Norm / manual
50	0.003602	1,625,625.0	0.000284	1,625,625.0	0.001235	1,625,625.0
50	0.003732	1,625,625.0	0.000292	1,625,625.0	0.001267	1,625,625.0
50	0.003618	1,625,625.0	0.000298	1,625,625.0	0.001206	1,625,625.0
50	0.003685	1,625,625.0	0.000284	1,625,625.0	0.001413	1,625,625.0
50	0.003320	1,625,625.0	0.000291	1,625,625.0	0.001307	1,625,625.0
50	0.003635	1,625,625.0	0.000306	1,625,625.0	0.001214	1,625,625.0
100	0.019787	25,502,500.0	0.002165	25,502,500.0	0.009916	25,502,500.0
100	0.019724	25,502,500.0	0.002610	25,502,500.0	0.009465	25,502,500.0
100	0.018439	25,502,500.0	0.002037	25,502,500.0	0.009821	25,502,500.0
100	0.018494	25,502,500.0	0.002125	25,502,500.0	0.010103	25,502,500.0
100	0.018573	25,502,500.0	0.002042	25,502,500.0	0.009436	25,502,500.0
100	0.018571	25,502,500.0	0.002355	25,502,500.0	0.009768	25,502,500.0
50	0.003495	1,625,625.0	0.000288	1,625,625.0	0.001215	1,625,625.0
50	0.003394	1,625,625.0	0.000284	1,625,625.0	0.001221	1,625,625.0
50	0.003525	1,625,625.0	0.000289	1,625,625.0	0.001229	1,625,625.0
50	0.003378	1,625,625.0	0.000280	1,625,625.0	0.001226	1,625,625.0
50	0.003452	1,625,625.0	0.000284	1,625,625.0	0.001432	1,625,625.0
50	0.003316	1,625,625.0	0.000289	1,625,625.0	0.001217	1,625,625.0
100	0.018835	25,502,500.0	0.002032	25,502,500.0	0.009974	25,502,500.0
100	0.019427	25,502,500.0	0.002039	25,502,500.0	0.009459	25,502,500.0
100	0.018434	25,502,500.0	0.002067	25,502,500.0	0.009416	25,502,500.0
100	0.019503	25,502,500.0	0.002044	25,502,500.0	0.009552	25,502,500.0
100	0.019931	25,502,500.0	0.002032	25,502,500.0	0.009669	25,502,500.0
100	0.018945	25,502,500.0	0.002147	25,502,500.0	0.009493	25,502,500.0

./A4-mpi-1D-cblas -r 50 → 1000

|Matrix|	Time / mpi	Inf Norm / mpi	Time / dgemm	Inf Norm / dgemm	Time / manual	Inf Norm / manual
50	0.003295	89,502.0	0.000282	89,502.0	0.001214	89,502.0
50	0.003377	89,502.0	0.000293	89,502.0	0.001245	89,502.0
50	0.003426	89,502.0	0.000289	89,502.0	0.001233	89,502.0
100	0.018321	353,266.0	0.002047	353,266.0	0.009422	353,266.0
100	0.023710	353,266.0	0.002265	353,266.0	0.009528	353,266.0
100	0.018545	353,266.0	0.002035	353,266.0	0.009973	353,266.0
500	3.974433	8,117,937.0	0.852176	8,117,937.0	3.526722	8,117,937.0
500	3.781977	8,117,937.0	0.901864	8,117,937.0	3.478443	8,117,937.0
500	3.660107	8,117,937.0	0.813615	8,117,937.0	3.387792	8,117,937.0
500	3.669998	8,117,937.0	0.826524	8,117,937.0	3.432746	8,117,937.0
1000	25.021923	31,896,067.0	6.877664	31,896,067.0	23.402237	31,896,067.0
1000	25.668985	31,896,067.0	6.845725	31,896,067.0	24.111009	31,896,067.0
1000	25.339146	31,896,067.0	7.069291	31,896,067.0	24.148800	31,896,067.0
1000	25.378952	31,896,067.0	6.827083	31,896,067.0	24.631762	31,896,067.0
50	0.003793	89,502.0	0.000300	89,502.0	0.001219	89,502.0
50	0.003746	89,502.0	0.000614	89,502.0	0.001306	89,502.0
50	0.003574	89,502.0	0.000284	89,502.0	0.001208	89,502.0
100	0.019365	353,266.0	0.002036	353,266.0	0.009537	353,266.0
100	0.022317	353,266.0	0.002033	353,266.0	0.009473	353,266.0
100	0.018096	353,266.0	0.002620	353,266.0	0.015792	353,266.0
500	3.794989	8,117,937.0	0.823255	8,117,937.0	4.208720	8,117,937.0
500	3.594502	8,117,937.0	0.872222	8,117,937.0	3.590184	8,117,937.0
500	3.834893	8,117,937.0	0.837537	8,117,937.0	3.589217	8,117,937.0
500	3.588232	8,117,937.0	0.815256	8,117,937.0	3.826735	8,117,937.0
1000	23.135391	31,896,067.0	6.489187	31,896,067.0	22.191726	31,896,067.0
1000	23.256875	31,896,067.0	6.380877	31,896,067.0	22.198515	31,896,067.0
1000	23.812851	31,896,067.0	6.490945	31,896,067.0	22.314222	31,896,067.0
1000	23.409070	31,896,067.0	6.580239	31,896,067.0	22.907298	31,896,067.0

./A4-mpi-1D-cblas -i 50 → 1000

|Matrix|	Time / mpi	Inf Norm / mpi	Time / dgemm	Inf Norm / dgemm	Time / manual	Inf Norm / manual
50	0.003397	1,625,625.0	0.000288	1,625,625.0	0.001242	1,625,625.0
50	0.003611	1,625,625.0	0.000287	1,625,625.0	0.001215	1,625,625.0
50	0.003682	1,625,625.0	0.000288	1,625,625.0	0.001211	1,625,625.0
100	0.019267	25,502,500.0	0.002433	25,502,500.0	0.009435	25,502,500.0
100	0.018404	25,502,500.0	0.002047	25,502,500.0	0.009442	25,502,500.0
100	0.023596	25,502,500.0	0.002019	25,502,500.0	0.012373	25,502,500.0
500	3.669566	15,687,562,500.0	0.818055	15,687,562,500.0	3.356318	15,687,562,500.0
500	3.575005	15,687,562,500.0	0.821899	15,687,562,500.0	3.329264	15,687,562,500.0
500	3.580136	15,687,562,500.0	0.827713	15,687,562,500.0	3.289661	15,687,562,500.0
500	3.638119	15,687,562,500.0	0.810202	15,687,562,500.0	3.298383	15,687,562,500.0
1000	23.389160	250,500,250,000.0	6.553018	250,500,250,000.0	23.616626	250,500,250,000.0
1000	23.609556	250,500,250,000.0	6.560736	250,500,250,000.0	22.512074	250,500,250,000.0
1000	23.316519	250,500,250,000.0	6.552578	250,500,250,000.0	22.917504	250,500,250,000.0
1000	23.223686	250,500,250,000.0	6.693635	250,500,250,000.0	22.248374	250,500,250,000.0
50	0.004897	1,625,625.0	0.000281	1,625,625.0	0.001219	1,625,625.0
50	0.003326	1,625,625.0	0.001049	1,625,625.0	0.002203	1,625,625.0
50	0.003305	1,625,625.0	0.000286	1,625,625.0	0.001210	1,625,625.0
100	0.018313	25,502,500.0	0.002023	25,502,500.0	0.009471	25,502,500.0
100	0.018587	25,502,500.0	0.002033	25,502,500.0	0.009530	25,502,500.0
100	0.019000	25,502,500.0	0.002073	25,502,500.0	0.009658	25,502,500.0
500	3.710844	15,687,562,500.0	0.834674	15,687,562,500.0	3.355016	15,687,562,500.0
500	3.587707	15,687,562,500.0	0.819234	15,687,562,500.0	3.319822	15,687,562,500.0
500	3.619530	15,687,562,500.0	0.816740	15,687,562,500.0	3.347187	15,687,562,500.0
500	3.604923	15,687,562,500.0	0.818356	15,687,562,500.0	3.359344	15,687,562,500.0
1000	23.619057	250,500,250,000.0	6.392954	250,500,250,000.0	21.995911	250,500,250,000.0
1000	24.271146	250,500,250,000.0	6.441605	250,500,250,000.0	22.635459	250,500,250,000.0
1000	23.330612	250,500,250,000.0	6.564853	250,500,250,000.0	22.277569	250,500,250,000.0
1000	22.908853	250,500,250,000.0	6.467756	250,500,250,000.0	22.457247	250,500,250,000.0

Appendix III – References / Acknowledgements

- www.stackoverflow.com : general queries on MPI function not working and possible workaround
- http://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms
- http://en.wikipedia.org/wiki/Message_Passing_Interface
- www.emsl.pnl.gov : Pacific Northwest National Laboratory / Batelle : "Introduction to MPI"
- www.ucd.ie : COMP40700 High Performance Computing – Notes 2014 "Message Passing Libraries"