COMP-40730 HPC

Report for Assignment 4

Author:	Paula Dwan
Due date:	30-June-2014
Lecturer:	Alexey Lastovetsky
Subject:	COMP-40730 High Performance Computing
College:	University College Dublin

Contents

EXERCISE

Write a parallel MPI program computing the product of two n×n dense matrices on p processors 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 utilized MPI when calculating manually and BLAS when calculating otherwise. :

- **A4-mpi.c**

Algorithm

Assignment Execution

Each program was executed multiple times using the script *./runAssignment4.sh*. This has multiple options and the syntax and usage follows:

Execute this script in the home directory of Assignment 4, 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-A4-mpi-<current time>-values<iteration>.txt`
example:	`pdwan-A4-mpi-20140628-022426-values-0.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-A4-mpi-timing<iteration>.dat`
example:	`pdwan-A4-mpi-20140628-022426-timing-0.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 :

Sample .dat file

```
# Program running :    A4-mpi.c
# File name is:        pdwan-A4-mpi-20140628-022426-timing-0.dat
#
# Computation of infinity norm of matrix C B using openMP
#
# 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.

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.`
	
	`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 MPI”`
	`set xlabel “Time taken / ms”`
	`set ylabel “size of block / matrix”`
	`plot`
	`    'pdwan-A4-mpi-20140628-022426-timing-0.dat'`
	`        u 1:3 t 'Matrix : manual' w l lw 0.5 lc rgb 'blue'`
	
	`    'pdwan-A4-mpi-20140628-022426-timing-0.dat'`
	`        u 2:3 t 'Block: manual' w l lw 0.5 lc rgb 'green'`
	
	`    'pdwan-A4-mpi-20140628-022426-timing-0.dat'`
	`        u 1:4 t 'Matrix : dgemm' w l lw 0.5 lc rgb 'black'`
	
	`    'pdwan-A4-mpi-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 :

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

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.