Final Lab Assignment-cum-Evaluation

22th November, 2016

High Performance Computing - CS301
DA-IICT, Gandhinagar, India

1 Instructions

This final lab assignment is aimed at collecting reproducible performance data and evaluating your ability to **interpretat the performance trends/results**.

You will be assigned one problem to perform in this lab. This will be a problem that you have already worked on in one of the previous assignments. You will also be assigned a particular approach to be used to solve the problem. You will be required to do the following:

1.1 Before the Lab

- Write both the serial and the parallel code and submit the same by Tuesday morning 9:00 AM.
  (Details in Section 2)
- Prepare to explain the results that you will obtain. You will perform this analysis in the lab itself. No submission of analysis required beforehand.
  (Details in Section 5)

1.2 During the Lab

- Run the codes that you have submitted using the scripts that you have been provided and submit the results that you obtain. Also, submit the hardware details in a format that will be specified during the lab hours.
- Analyze the results that you obtain, according to the points mentioned in Section 5.
- Perform task 2, details of which will be disclosed during the lab hours.

There will be two major components on which you will be evaluated. The first will be the clarity (readability) and conciseness (without unnecessary overheads) of your code and the comments in your code. The second will be the analysis that you provide for whatever results you obtain. You will not be analysed on the basis of the speedup you get, therefore you don't need to be distressed if you think that the particular approach you have been assigned is a poor approach. It is your job to analyse the results you obtain in a comprehensive manner.

2 Codes

For the problem and approach you have been assigned you are required to submit 4 codes: two serial codes and two parallel codes. In both serial and parallel, one of the codes should contain timers and print statements that calculate and print the times required for running the code while the other code should not have any such function calls/statements that are not required to solve the problem.

- serial-measurement : Serial code. Contains timers and print statements.
- serial-clean : Serial code. No extraneous statements.
- parallel-measurement : Parallel code. Contains timers and print statements.
- parallel-clean : Parallel code. No extraneous statements.

All codes should take in command line arguments in the following manner: `./a.out n p [input-file]`
Here, "a.out" is the name of your executatble file, "n" is the problem size, "p" is the number of processors (keep $p = 0$ for serial code) and "input-file" is an optional argument[1] which is the name of the file from which input is to be taken in. The details of the input and output file formats are mentioned in Section 4.

For uniformity, a script (`create_input.py`) will be provided that produces the input for the problems that require an input. You will require certain Python3 modules (*numpy*, *scipy*, *matplotlib*, *pandas*) for the same. In case you do not have these installed, follow the installation instructions mentioned in Appendix A. Further, for ease of evaluation, a script (`run.py`) will be provided that that will be used by us to run your codes for different problem sizes and for different number of processors. You will also be provided a script (`plotter.py`) that computes the basic metrics and plots the graphs for you to analyse your code. You are encouraged to use these scripts to test your code before submitting.

Each of the 4 codes that you submit should be properly commented; it should be possible for anyone familiar with the basics of C and HPC to read the code and gain a good understanding of how the code works. At the same time, do not clutter the code with unnecessary comments. You will be evaluated on the clarity and conciseness of the code and comments; excessive verbosity is discouraged.

3 Approaches

The problem that you have been assigned are the same as those that you have done during your assignments. The various approaches that you have been assigned are explained in this section. You are required to stick to the approach and any optimizations you perform need to be in such a manner that the approach does not change in the process. The aim of this assignment is to evaluate your ability to interpret results/trends not to evaluate your ability to optimize approaches to solving a problem.

3.1 Problem: Trapezoidal

Reduction: This approach requires you to use the reduction directive of OpenMP as applicable.

Critical: This approach requires you to use the critical directive of OpenMP as applicable.

Array based: This approach requires you to use the an array of variables to store intermediate sums.

3.2 Problem: Vector Multiplication

Static Scheduling: Use the static scheduling of OpenMP.

Dynamic Scheduling: Use the dynamic scheduling of OpenMP.

Two elements together: This approach requires you change the two vectors and make a new array so that corresponding elements in the two original arrays are adjacent in the new array. The problem is then solved using this array.

3.3 Problem: Matrix Multiplication

Outermost Loop: Parallelize the outermost loop.

[1] The "input-file" argument is not necessary for the Trapezoidal, Pi using Series and Monte Carlo problems.

Middle Loop: Parallelize the middle loop.

Use B Transpose: Take the transpose of the second matrix before multiplying.

Block Matrix: Parallelize the block matrix multiplication method.

3.4 Problem: Pi using Series

Use the pow() function: Call the pow() function in your code to solve the problem.

Critical: This approach requires you to use the critical directive of OpenMP as applicable.

Reduction: This approach requires you to use the reduction directive of OpenMP as applicable.

3.5 Problem: Image Warping

Data division: Divide the image data into chunks, using the outer loop.

Collapsed Directive: This approach requires you to use the collapsed directive of OpenMP as applicable.

3.6 Problem: Median Filtering

Inbuilt qsort: Use the inbuilt qsort function of C while finding median. The rest of the approach is normal image division into chunks.

Different Memory Allocation: This approach requires you to use a different memory allocation scheme for the problem.

3.7 Problem: Monte Carlo

Using rand(): Use the rand() function.

Using rand_r() with reduction: Use the rand_r() function along with the reduction directive of OpenMP as applicable.

Using rand_r() with critical: Use the rand_r() function along with the critical directive of OpenMP as applicable.

Using your own PRNG: Make your own PRNG (say `xorshift`) along with the reduction directive of OpenMP as applicable.

3.8 Problem: Reduction

Tree: Use the tree approach.

Data Segmenting: Segment the data into chunks, perform reduction on each and then combine.

3.9 Problem: Prefix Sum

Double Tree: Use the double tree approach.

Data Segmenting: Segment the data into chunks, perform prefix sum on each and then combine.

3.10 Problem: Filtering

Using double tree prefix sum: Use the double tree approach for prefix sum and use that for filtering.

Using Data Segmenting prefix sum: Use the data segmenting approach for prefix sum and use that for filtering.

Using Linked List: Use a linked list implementation for ensuring that the elements are filtered in order.

4 Input and Output File Formats

4.1 Matrix Multiplication

You are supposed to be multiply two square matrix of dimensions (n,n) and (n,n). Each element of the input file will be a double in the range [0,1)

Input Format:
The program should take three command line arguments. The first would be n, which is the dimension of matrix. The second would be p, which is the number of threads which will be launched. The third would be the name of the file (which we will generate for you) from which you will read the input.

Output File Format:
The output should be written to a file where each element of each row should be separated by a space and rounded off to 6 decimal places. Each row should be in a new line.

Example Input/Output Files
If you are to multiply two matrix A and B of dimensions (3,3), then the input file would look like this. It would have 6 lines(3+3).

```
0.001 0.003 0.004
0.005 0.006 0.007
0.008 0.008 0.009
1.000 0.000 0.000
0.000 1.000 0.000
0.000 0.000 1.000
```

Here A is the matrix

```
0.001 0.003 0.004
0.005 0.006 0.007
0.008 0.008 0.009
```

And B is the matrix

```
1.000 0.000 0.000
0.000 1.000 0.000
0.000 0.000 1.000
```

The output file should look like this (it will have 3 lines). Note that exactly 6 decimal places' precision is required.

```
0.001000 0.003000 0.004000
0.005000 0.006000 0.007000
0.008000 0.008000 0.009000
```

4.2 Prefix Sum

You are supposed to calculate the prefix sum of a given array. Each element of the input array would be an integer between [0,10).

Input Format:
The program should take three command line arguments. The first would be n, which is the number of elements of the array. The second would be p, which is the number of threads which will be launched. The third would be the name of the file (which we will generate for you) from which you will read the input.

Output File Format:
The output should be written to a file where each element should be separated by a space.

Example Input/Output Files
If you have to do the prefix sum of an array A, there would be a single line in the input file.

```
1 2 3 4 5 6 7 8 9 10
```

The output file should look like this(It will have a single line) and each element separated by a space.

```
1 3 6 10 15 21 28 36 45 55
```

4.3 Reduction

You are supposed to apply the reduction with the binary operator as the addition operator on a given array. Each element of the input array would be an integer between [0,10).

Input Format:
The program should take three command line arguments. The first would be n, which is the number of elements of the array. The second would be p, which is the number of threads which will be launched. The third would be the name of the file (which we will generate for you) from which you will read the input.

Output File Format:
The output should be written to a file where there will be only one integer which is the sum of all the elements of the array.

Example Input/Output Files
If you have to do the prefix sum of an array A, there would be a single line in the input file.

```
1 2 3 4 5 6 7 8 9 10
```

The output file should look like this(It will have a single line) and a single integer.

```
55
```

4.4 Vector Multiplication

You are supposed to calculate the dot product of two given vectors. Each element of the input array would be an integer between [0,10).

Input Format:
The program should take three command line arguments. The first would be n, which is the number of elements in the vector. The second would be p, which is the number of threads which will be launched. The third would be the name of the file (which we will generate for you) from which you will read the input.

Output File Format:
The output should be written to a file where there will be only one integer which is the dot product of two vectors.

Example Input/Output Files
If you have to do the dot product of two vectors A and B, it will have two lines of input. Each element in each line will be separated by a space.

```
1 2 3 4 5 6 7 8 9 10
1 1 1 1 1 1 1 1 1 1
```

The output file should look like this(It will have a single line) and a single integer.

```
55
```

4.5 Filtering

You are supposed to generate an array of elements which are greater than a given value K(which is 4 in this case) from a given array.

Input Format:
The program should take three command line arguments. The first would be n, which is the number of elements in the array. The second would be p, which is the number of threads which will be launched. The third would be the name of the file (which we will generate for you) from which you will read the input.

Output File Format:
The output file will have a single line. They will be the elements which satisfy the given condition.

Example Input/Output Files
There will be one integer in the first line containing the threshold element. This will be followed by a single line and each of the element of the array would be separated by spaces.

```
4
4 4 4 4 5 5 6 1 4 4 4 9 9 9
```

The output file should look like this(It will have a single line) and a single integer.

```
5 5 6 9 9 9
```

4.6 Image Processing Based Assignments

You are supposed to the median filtering/Image Warping. The image file will be in a ppm format and will be square.

Input Format:
The program should take three command line arguments The first would be n, the size of one of the dimensions of the image The second would be p, which is the number of threads which will be launched The third would be the name of the file (which we will generate for you) from which you will read the input.

Output File Format:
The output file will be a file which will have the warped/filtered image.

Input/Output Files
The input file will be a ppm file and the output file should also be a ppm file.

4.7 Trapezoidal/Pi using Series/Monte Carlo

Trapezoidal is the problem where you compute the integral that you computed in your assignment. Pi using series is the problem requires you to compute the value of Pi using the series from your earlier assignment. Monte Carlo is the problem You are supposed to the median filtering/Image Warping. The image file will be in a ppm format and will be square.

Input Format:
The program should take three command line arguments The first would be n, the size of one of the dimensions of the image The second would be p, which is the number of threads which will be launched The third would be the name of the file (which we will generate for you) from which you will read the input.

Output Format:
The output file will be a file which will have the warped/filtered image.

Input/Output File
The output file should contain one element containing the value computed in your problem.

5 Analysis Expected

The analysis that you provide during the lab hours is a key component of your evaluation and thus it needs to be accurate as well as comprehensive. The following points should be kept in mind. The exact questions may be slightly different and you will be required to answer them on the day of the lab.

- The shape of the speedup curve, particularly, the reasons for trends like increase/decrease, saturation, etc.
- Explanation in terms of compute and memory accesses that the code makes. These include the importance of compute to memory accesses ratio as well as concepts like memory walls.
- Speedup bounds that various metrics provide you with. You should be prepared to answer questions regarding how Amdahl's law/effect, Gustafson-Barsis law, Karp-Flatt Metric and the Isoefficienc metric apply to your particular problem/approach combination.
- Role of cache(s) in the trends that are observed. These include cache coherence, false sharing as well as order of memory access.
- Synchronization overheads that may be present.
- The effect of thread scheduling.
- Granularity of the algorithm or algorithm sub-steps.
- Miscellaneous other concepts that may apply in your particular problem/approach combination. Things like loop unrolling, function call overheads, thread safe functions, etc. should also be mentioned in your analysis.

You will also be required to provide some commentary on the quality of the approach for the particular problem, its advantages/disadvantages and any difficulties that are faced while parallelizing using that particular approach.

Apart from this you will be required to provide the performance measured in memory accesses and FLOPS. You can refer to this simple example table for the number of FLOPS you need to count for these basic operations.

Table 1. Basic instructions alongside the bytes accessed and FLOPS. The last row has 3 loads and 1 store in it.

Name	Instruction	Bytes/Iteration	FLOPS/Iteration
COPY	a(i) = b(i)	16	0
SCALE	a(i) = q*b(i)	16	1
SUM	a(i) = b(i) + c(i)	24	1
TRIAD	a(i) = b(i) + c(i)*d(i)	32	2

A Python Instructions

You will need to install `numpy, scipy, pandas and matplotlib` to run the scripts that we will provide. Instructions for installation are as follows.

A.1 Linux installation from repository

If you are using a recent version of Ubuntu (14.04 or later) or Arch Linux (or many other Linux distros), you can install the following packages using your package manager: python3, python3-numpy, python3-scipy, python3-matplotlib, python3-pandas.

A.2 Linux installation using pip

Any recent Linux distribution should provide you python3 in the repos. You could then try and install pip for python3. If you are able to do that, you can try `PYTHON3 -m pip install numpy`, where PYTHON3 is the path to the Python3 that you installed (it will just be python3 if it is somewhere on the executables path of your system). You can do the same for scipy, matplotlib and pandas. This may not be so simple because you will first need to install a variety of development tools including gcc, gfortran and development versions of Atlas/BLAS/LAPACK using your package manager. If you want to install for all users you will need a sudo in front: `sudo PYTHON3 -m pip install numpy` (keep in mind that even if python3 is in your path, it need not be in root's path and PYTHON3 will often need to be the full path whenever you use sudo). If you find it difficult to make all this work, you probably want to go the Anaconda route discussed later.

A.3 Easy installation on Windows, Linux or Mac

The most feasible way of installing Python3/Numpy/Scipy on Windows or Mac is to use Anaconda; on Linux, you have other options as well (discussed below), but Anaconda is still a reasonable choice. Download the version for your platform and follow the instructions at the above link. This is a huge download (somewhat less than half a GB) and the installed size is around 2 GB but Anaconda is a comprehensive collection of Python modules that will come in handy in other situations as well. The instructions on the Anaconda website will install it only for you in your home folder. If you want to install it for all users, you could do something like:

```
sudo ./Anaconda3-4.1.1-Linux-x86_64.sh -p /opt/anaconda3
## Python 3.5 is then available as "/opt/anaconda3/bin/python3"
## and you could use this path wherever I refer to PYTHON3 below
## You could also add "/opt/anaconda3/bin/" to the PATH
```

You could also try other distributions like Enthought.

Note: This document, in its original form, was provided to us by Prof. Jayanth Varma and Prof Vineet Virmani as a part of CS-401 Computational Finance.