COMP3001
Parallel Programming

20 CREDIT MODULE

ASSESSMENT: 100% Coursework **W1: 30% Set Exercises**
W2: 70% Report

MODULE LEADER: Dr. Vasilios Kelefouras

MODULE AIMS

This module aims on giving students the practical understanding and skills needed to parallelize and optimize real world programs on modern processors. Students will learn about state of the art high performance computing (HPC) technologies, hardware computer architectures and parallel programming models. This module will give students real world coding experience and hands on project work.

ASSESSED LEARNING OUTCOMES (ALO):

1. Understand state of the art high performance computing technologies and parallel programming models
2. Write performance efficient code to speedup programs on modern multi-core processors
3. Design and implement performance efficient programs on GPUs.

Overview

This document contains all the necessary information pertaining to the assessment of *COMP3001 Parallel Programming*. The module is assessed via **100% coursework**, across two elements: *30% Set Exercises* and *70% Report*.

The sections that follow will detail the assessment tasks that are to be undertaken. The submission and expected feedback dates are presented in Table 1. All assessments are to be submitted electronically via the respective DLE module pages before the stated deadlines.

	Submission Deadline	Feedback
Set Exercises (30%)	**29th Nov. 2022 3pm**	Within 20 working days
Report (70%)	**16th Jan. 2023 3pm**	Within 20 working days

Table 1: Assessment Deadlines

All assessments will be introduced in class to provide further clarity over what is expected and how you can access support and formative feedback prior to submission. Whilst the assessment information is provided at the start of the module, it is not necessarily expected you will start this immediately – as you will often not have sufficient understanding of the topic. The module leader will provide guidance in this respect.

Assessment 1: Set Exercises (30%)

The set of exercises consists of the following two questions. All the source files needed can be found here https://github.com/kelefouras/comp3001/tree/newversion/comp3001-master/coursework . You need to provide information about the hardware you run the experiments.

Question 1 – Run an Experimental Procedure and Analyze the Results

Download the 'q1a.c' and 'q1b.c' source code files from the module's GitHub repository. Note that each file has two versions, one for Visual Studio and another for Linux; download the right file based on the environment you want to work on. The source code files include a computationally expensive block of code and a timer that measures its execution time.

Do the following tasks:

A. Run the 'q1a.c' program for the following five input sizes '*N=[4096,4096*2,4096*4,4096*16,4096*32]'* and measure the FLOPs (floating point operations per second) values being achieved. Make a graph of '*FLOPs vs N'*. Then, run the 'q1b.c' program for the following input sizes '*N=[256,512,1024,2048,4096,8192,16384]'* and measure the FLOPs values being achieved. Make a graph of '*FLOPs vs N*'. Note that in 'q1b.c' program, you must calculate and print the FLOPs value on your own. [15 marks]

Tip. To get an accurate execution time value, the program must run for at least 20 seconds. If you measure the FLOPs value in a virtual machine, then make sure this is clear on the graph's title, as the execution time will be much higher in this case (it is preferable to measure the execution time in a non-virtual machine environment).

Marks	**0-5**	**6-15**
Marking Criteria	The student has either provided incomplete figures (where some values are missing) or some values are wrong.	The student has provided figures where all the FLOPs values are shown in a clear way. The experimental procedure has been well carried out.

B. Briefly explain the graph results found in step A. What is the theoretical peak FLOPs value on your computer? Why the measured value is lower than that? [15 Marks]

Marks	**0-3 marks**	**4-7**	**8-15**

Marking Criteria	Little or no analysis. Little understanding of the subject. Almost no explanation is given	Answers are given at a good level of detail and are explained clearly. The student has explained how the FLOP values are affected by the input size and why.	An outstanding analysis is provided. The student has explained why the measured FLOPs values are lower than the theoretical peak ones. An outstanding analysis is provided.

C. Use Cachegrind tool (Valgrind) to measure the number of L1 data cache misses for '*q1b.c*' program when '*N=4096'* (set the TIMES variable to one - TIMES=1). Theoretically justify the result. [10 Marks]

Marks	**0-4**	**5-10**
Marking Criteria	The student has not well measured the number of misses. Little or no analysis. Little understanding of the subject.	An outstanding analysis is provided. The student justifies the number of misses measured, by analyzing the data access patterns and the cache size. An outstanding analysis is provided.

Question 2 – Speeding up a linear algebra application on your PC

Drawing upon all the optimization techniques that you have learned so far in this module, you will speedup the '*slow_routine()*' located in 'q2.*c*' file. The main optimization needs to be applied is vectorization, by using ***x86-64 SSE/SSE2/SSE4/AVX/AVX2*** C/C++ intrinsics. Provide the FLOPs value being achieved before and after optimization. All the C/C++ x86-64 intrinsics are provided in the following link: https://software.intel.com/sites/landingpage/IntrinsicsGuide/ .

The marking scheme is as follows **[60 marks]**:

Marks	**0-9 marks**	**10-25 marks**	**26-45 marks**	**46-60 marks**
Marking criteria	The student has not provided appropriate vectorised code. The routine is not fully vectorised. The routine does not generate the right output.	The routine is fully vectorised. However, the implementation contains bad practice. The implementation does not work for any input size. The flops value being achieved before and after the optimization is provided.	Vectorization has been efficiently applied. Register blocking has been efficiently applied.	An outstanding implementation is provided. Other optimizations are provided too, such as loop tiling.

***Hints**: There are many different ways to implement this routine and each solution includes different intrinsics. However, a valid solution exists by using the following SSE instructions: _mm_load_ps, _mm_load_ps1, _mm_load_ss, _mm_add_ps, _mm_mul_ps, _mm_store_ps, _mm_store_ss, _mm_set1_ps, _mm_hadd_ps.*

Submission Details

The submission will be done via the submission link on the COMP3001 DLE page. You should submit **two files**:

1. A q1.docx or q1.odt file containing the solution of question 1.
2. The q2.c or q2.cpp file containing the solution of question 2.

Do **not** upload any .zip files.

Assessment 2: Report (70%)

Question 1 – Parallelize a software application using OpenMP framework [40 Marks]

Your task is to parallelize a C software application by using OpenMP framework. You can find the code on the module's GitHub repository. You can perform this task either in Linux or in Visual Studio 2019. The marking criteria are as follows.

Marks	0-7	8-18	19-29	30-40
Marking Criteria	The student has not used the OpenMP annotations appropriately. The student has not used the OpenMP annotations to all the loop kernels that can be parallelized*. The program does not generate the right output.	The student has used the OpenMP annotations appropriately, but just for a few loop kernels. He/she has not used the OpenMP annotations to all the loop kernels that can be parallelized*.	The student has used the OpenMP annotations appropriately to all the loop kernels that can be parallelized*. However, the code delivered includes either multi-threaded code only or vectorized code only, not both.	The student has used the OpenMP annotations appropriately for all the loop kernels that can be parallelized*. The code delivered contains both multi-threaded and vectorized code.

* *If a loop kernel cannot be parallelized or vectorized in its current form, then you do NOT have to take actions against this problem, e.g., the loop kernel in line 178 cannot be vectorized by using OpenMP (in its current form).*

***Hint**: The Openmp clauses that need to be used are the following: omp for, reduction, private, shared, omp simd, omp for simd, aligned. Note that the aligned clause requires that the_mm_malloc should be used instead of malloc.*

Question 2 – CPU vs GPU performance evaluation and analysis [35 Marks]

You are provided with three different implementations of Matrix-Vector Multiplication (MVM) algorithm, two for the CPU and one for the GPU; these implementations are shown in the q2.cu file in the module's GitHub repository. The first two versions have been studied in the lab sessions (so the full code is provided there). You are not provided with a code template. It is part of this task to find and re-use the code that has been given in the lab sessions.

1. **Measure the FLOPs values of the three implementations. Make a single graph showing '*FLOPs vs N*'** for the three routines (the FLOPs values will be shown in the y-axis while the N values will be shown in the x-axis). Consider using a logarithmic scale in the y-axis. Use square matrices of size NxN, where *N=[256,512,1024,1024*2, 1024*4, 1024*8, 1024*16]*. The GPU implementation will run on the lab's GPUs. The two CPU implementations will run on any PC; however, it is strongly recommended to run them on a non-virtual machine environment, e.g., consider using the PCs in SMB201. If you use VS, make sure you run the codes under the 'release mode'. If you use Linux, compile the codes by using '-O3' option. When measuring the execution time on the GPUs, include the time needed to transfer the arrays from the CPU to the GPU and vice versa. **[10 marks]**

Marks	**0-5**	**6-10**
Marking Criteria	The student has either provided an incomplete figure (where some values are missing) or some values are wrong. The experimental procedure is not well applied.	The student has provided a figure where all the FLOP values are shown in a clear way. All the FLOPs values are correctly measured.

2. Explain the results you found in task1**. [10 marks]**

Marks	**0-2 marks**	**3-7**	**8-10**
Marking Criteria	Little or no analysis. Little understanding of the subject. Almost no explanation is given	Answers are given at an appropriate level of detail and are explained clearly. The student has explained how the FLOP values are affected by the input size and why, on both the CPU and the GPU	An outstanding analysis is provided. The text provided is well written.

3. Drawing upon the optimization techniques that you have learned in this module, you will speedup the CUDA MVM implementation provided above. The new CUDA kernel will be provided in a separate routine

entitled mvm_ver4(). Make the graph of FLOPS vs N, as in question 2.1. The marking scheme is as follows: **[15 marks]**

Marks	0-2	3-5	6-15
Marking Criteria	The student has **not** provided a parallel implementation that generates the right output.	Marginal speedup is achieved. The implementation contains bad practices.	The student has provided an outstanding implementation of high performance. Shared memory is used. The graph of FLOPS vs N is provided.

Question 3 – Optimize part of a Deep Neural Network application [25 Marks]

Drawing upon the optimization techniques that you have learned in this module, you will speed up part of a deep neural network application. Under the question3 folder in the module's GitHub repository, you will find a C application that executes the convolution and ReLU layers of a convolutional neural network (CNN). Note that the convolution layers are responsible for about 80%-90% of the overall execution time of modern CNNs. Therefore, their optimization is of paramount importance. You can use either Linux or Windows/Mac (Visual Studio). The routine to be optimized is the '*unoptimized_layer_FP()*', which is located in the *convolution_layer_2D.c* file. Do not optimize any other routines or block of code. Provide the speedup value achieved as well as the FLOPs achieved. Note that there is no single solution. **[25 marks]**

The marking criteria are as follows:

Marks	0-2	3-5	6-15	16-25
Marking Criteria	The code does not generate the right output. No optimization is applied.	The student has applied optimizations such as parallelization or register blocking. Vectorization is **not** applied by	The student has provided an efficient implementation. Optimizations include parallelization, SSE/AVX x86-64 intrinsics and register blocking.	The student has applied all the optimizations mentioned in the third column, in an efficient way. The student has provided the right SSE/AVX intrinsics and has applied them the right way.

		using SSE/AVX intrinsics.		

Tips:

1. *Try to understand how the algorithm works (Fig.1).*
2. *Consider vectorising either m or d loops. The latter is easier but gives lower performance.*

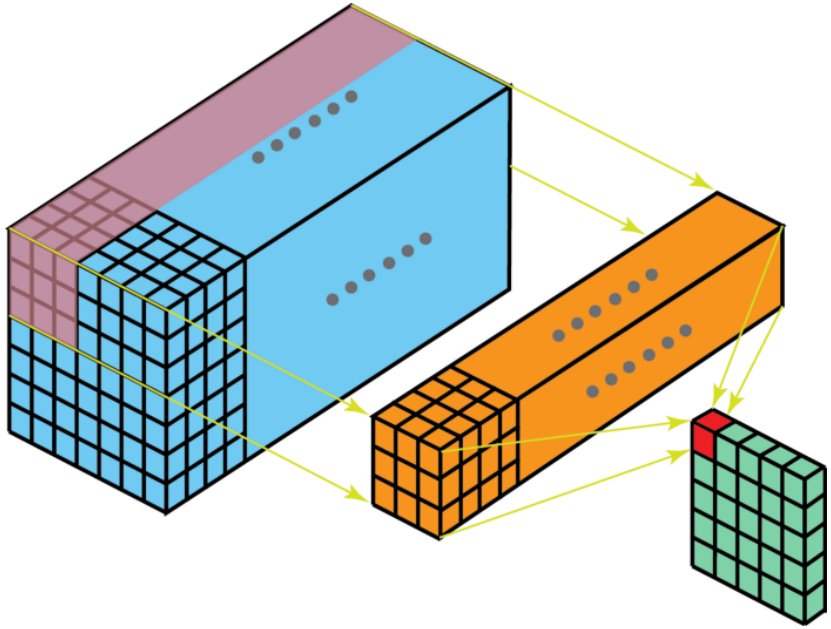

Fig.1 Visual illustration of the unoptimized_layer_FP() routine. In_FP[] array is shown on the left, while filter_FP[] array is shown in the middle.

Submission Details

The submission will be done via the submission link on the COMP3001 DLE page. You will submit **four files**:

- **q1.c** or **q1.cpp** file for question 1
- **q2.docx** file containing the answers for question 2.
- **q2.cu** file for question 2.3
- **q3.c or q3.cpp** file containing the solution of question 3

Do **not** upload any .zip files. Do **not** upload the whole Visual Studio Project.

General Guidance

Extenuating Circumstances

There may be a time during this module where you experience a serious situation which has a significant impact on your ability to complete the assessments. The definition of these can be found in the University Policy on Extenuating Circumstances here:
https://www.plymouth.ac.uk/uploads/production/document/path/15/15317/Extenuating_Circumstances_Policy_and_Procedures.pdf

Plagiarism

All of your work must be of your own words. You must use references for your sources, however you acquire them. Where you wish to use quotations, these must be a very minor part of your overall work.

To copy another person's work is viewed as plagiarism and is not allowed. Any issues of plagiarism and any form of academic dishonesty are treated very seriously. All your work must be your own and other sources must be identified as being theirs, not yours. The copying of another persons' work could result in a penalty being invoked.

Further information on plagiarism policy can be found here:

Plagiarism: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism

Examination Offences: https://www.plymouth.ac.uk/student-life/your-studies/essential-information/exams/exam-rules-and-regulations/examination-offences

Turnitin (http://www.turnitinuk.com/) is an Internet-based 'originality checking tool' which allows documents to be compared with content on the Internet, in journals and in an archive of previously submitted works. It can help to detect unintentional or deliberate plagiarism.

It is a formative tool that makes it easy for students to review their citations and referencing as an aid to learning good academic practice. Turnitin produces an 'originality report' to help guide you. To learn more about Turnitin go to:
https://guides.turnitin.com/01_Manuals_and_Guides/Student/Student_User_Manual

Referencing
The University of Plymouth Library has produced an online support referencing guide which is available here: http://plymouth.libguides.com/referencing.

Another recommended referencing resource is Cite Them Right Online; this is an online resource which provides you with specific guidance about how to reference lots of different types of materials.

The Learn Higher Network has also provided a number of documents to support students with referencing:
References and Bibliographies Booklet:
http://www.learnhigher.ac.uk/writing-for-university/referencing/references-and-bibliographies-booklet/
Checking your assignments' references:
http://www.learnhigher.ac.uk/writing-for-university/academic-writing/checking-your-assigments-references/