University of Wolverhampton
Faculty of Science and Engineering
Department of Mathematics and Computer Science

Module Assessment

Module	6CS005 High Performance Computing
Module Leader	Dr. Ali Safaa Sadiq
Semester	1
Year	2019/20
Assessment	Portfolio
% of module mark	100%
Due Date	23:59 17th January 2020
Hand-in – what?	Portfolio as specified in this document
Hand-in- where?	Canvas
Pass mark	40%
Method of retrieval	Resit. Specification of work to be done needs to be negotiated with module leader at feedback session
Feedback	Will be shared on Canvas for each submission
Collection of marked work	At feedback session

Assignment Overview:

This module is assessed by portfolio. Your portfolio will consist of:

- A learning journal that uses the specified template.
- Your source code and data files that use the specified structure.
- It is recommended that you use a revision control system, such as subversion, for maintaining your source code. This will enable your code to be backup up and will also provide a source of evidence in your defence if collusion is suspected.

The module is assessed by evidence that you provide of:

- Analysis of supplied programs and experimentation with them.
- Capturing results and depicting them as graphs.
- Written answers to the given related theoretical questions.
- Writing programs to solve computationally expensive problems that require High Performance Computing.
- Written explanations of your results and reflections of your learning.

Two different technologies, POSIX Threads and CUDA will be combined with one application domain of password cracking. The following sections are first ordered, by technology then by application domain.

Portfolio Tasks

Task 1 Parallel and Distributed Systems (15 Marks)

1. What are threads and what they are designed to solve? **(2 marks)**

2. Name and describe two process scheduling policies. Which one is preferable and does the choice of policies have any influence on the behavior of Java threads? **(2 marks)**

3. Distinguish between Centralized and Distributed systems? **(2 marks)**

4. Explain transparency in D S? **(2 marks)**

5. The following three statements contain a flow dependency, an antidependency and an output dependency. Can you identify each?

Given that you are allowed to reorder the statements, can you find a permutation that produces the same values for the variables C and B as the original given statements?

Show how you can reduce the dependencies by combining or rearranging calculations and using temporary variables. **(4 marks)**

Note: Show all the works in your report and produce a simple C code simulate the process of producing the C and B values. **(2marks for solving dependencies and 2marks for the code)**

B=A+C
B=C+D
C=B+D

6. What output do the following 2 programs produce and why? **(3 marks)**

#include <pthread.h> #include <stdio.h>	#include <pthread.h> #include <stdio.h>

int counter;
static void * thread_func(void * _tn) {
int i;
 	
for (i = 0; i < 100000; i++)
 	
counter++;
 	
return NULL;
 	
}
	int counter;
	static void * thread_func(void * _tn) {
	int i;
	
	for (i = 0; i < 100000; i++)
	
	counter++;
	
	return NULL;
	
	}

int main() {
 	
int i, N = 5;
 	
pthread_t t[N];
 	
for (i = 0; i < N; i++)
 	
pthread_create(&t[i], NULL, thread_func, NULL);
 	
for (i = 0; i < N; i++) pthread_join(t[i], NULL);
 	
printf("%d\n", counter);
 	
return 0;
 	
}
	int main() {
	
	int i, N = 5;
	
	pthread_t t[N];
	
	for (i = 0; i < N; i++) {
	
	pthread_create(&t[i], NULL, thread_func, NULL);
	
	pthread_join(t[i], NULL);
	
	}
	
	printf("%d\n", counter);
	
	return 0;
	
	}

Outputs: **(1.5 marks)**	Outputs: **(1.5 marks)**

Task 2: Applications of Matrix Multiplication and Password Cracking using HPC-based CPU system: (35 Marks)

Part A: Single Thread Matrix Multiplication (5 Marks)

Study the following algorithm that is written for multiplying two matrices A and B and storing the result in C.

```
int A[N][P], B[P][M], C[N][M];
for (int i = 0; i < N; i++)
    {
     for (int j = 0; j < M; j++)
       {
     C[i][j] = 0;
          for (int k = 0; k < P; k++)
          {
           C[i][j] = C[i][j] + A[i][k] * B[k][j];
          }
       }
     }
```

Now answer each of the following questions:

- Analyse the above algorithm for its complexity. **(1 marks)**
- Suggest at least three different ways to speed up the matrix multiplication algorithm given here. (Pay special attention to the utilisation of cache memory to achieve the intended speed up). **(1 marks)**
- Write your improved algorithms as pseudo-codes using any editor. Also, provide a reasoning as to why you think the suggested algorithm is an improvement over the given algorithm. **(1 marks)**
- Write a C program that implements matrix multiplication using both the loop as given above and the improved versions that you have written. **(1marks)**
- Measure the timing performance of these implemented algorithms. Record your observations. (Remember to use large values of N, M and P – the matrix dimensions when doing this task). **(1 marks)**

Part B: Write a code to implement matrix multiplication using multithreading (10 Marks)

- Write a C program to implement matrix multiplication using multithreading. The number of threads should be configurable at run time, for example, read via an external file. **(5 marks)**
- The code should print the time it takes to do the matrix multiplication using the given number of threads by averaging it over multiple runs. **(2.5 marks)**
- Plot the time it takes to complete the matrix multiplication against the number of threads and identify a sweet spot in terms of the optimal number of threads needed to do the matrix multiplication. **(2.5 marks)**
  (In doing this exercise, you should use matrices of large sizes e.g. 1024 * 1024 sized matrices).

Part C: Password cracking using POSIX Threads (20 Marks)

You will be provided with a template code of encrypting a password using SHA-512 algorithm. You are asked to run a code using the given template to encrypt a password contains **TWO** uppercase letters and **TWO** integer numbers **(total 4 digits)**. Afterwards, you need to run the given code to crack the password, i.e. find the plain-text equivalents. An example password is *AS12*.

1. Run the given cracking program 10 times and calculate the mean running time (Using 2 uppercase letters and 2 integer numbers). **(2 marks)**
2. In your learning journal make an estimate of how long it would take to run on the same computer if the number of initials were increased to 3. Include your working in your answer. **(2 marks)**
3. Modify the program to crack the three-initials-two-digits password given in the

three_initials variable. An example password is HPC19. **(2 marks)**

4. Write a short paragraph to compare the running time (average of 10 times) of your three_initials program with your earlier estimate. If your estimate was wrong explain why you think that is. **(2 mark)**
5. Modify the original version of the program to run on 2 threads. It does not need to do anything fancy, just follow the following algorithm. **(10 marks)**
   - Record the system time a nano second timer
   - Launch thread_1 that calls kernel_function_1 that can search for passwords starting from A-M
   - Launch thread_2 that calls kernel_function_2 that can search for passwords starting from N-Z
   - Wait for thread_1 to finish
   - Wait for thread_2 to finish
   - Record the system time using a nano second time and print the elapsed time.
6. Compare the results of the mean running time of the original program, (obtained by step no. 1), with the mean running time of the multithread version (10 running times). **(2 marks)**

Task 3: Applications of Password Cracking and Image Blurring using HPC-based CUDA system (50 Marks)

Part A: Password cracking using CUDA (25 Marks)

Using the same concept as before, you will now crack passwords using CUDA. Your program will take in an encrypted password and decrypt using many threads. CUDA allows multidimensional thread configurations so your kernel function (which runs on the GPU) will need to be modified according to how you configure the execution command.

Crack passwords using CUDA **(20 marks)**

Comparison, analysis and evaluation of CUDA version for password cracking (compare with normal and "pthread" version) **(5 marks)**

Part B: Image blur using multi dimension gaussian matrices (multi-pixel processing) (25 Marks)

Your program will decode a PNG file into an array (or 2D array) and apply the gaussian blur filter. Blurring an image reduces noise by taking the average RGB values around a specific pixel and setting its RGB to the mean values individually. For this assessment, you will use a 3x3 matrix however, higher marks will be given to those who allow the user to specify the blur matrix dimensions. This could be 5x5, 7x7, 9x9 etc.

Here is an example of how the blur works:

Suppose you have a 5x5 image such as the following (be aware that the coordinate values may change how you format your 2-dimensional array):

0,4	1,4	2,4	3,4	4,4
0,3	1,3	2,3	3,3	4,3
0,2	1,2	2,2	3,2	4,2
0,1	1,1	2,1	3,1	4,1
0,0	1,0	2,0	3,0	4,0

The shaded region above represents the matrix blur position, the yellow highlighted pixel is the one we want to blur, in this case, we are focusing on pixel 1,2 (x,y) (Centre of the matrix). To apply the blur for this pixel, you would sum all the **Red** values from the surrounding coordinates including 1,2 (total of 9 R values) and find the average (divide by 9). This is now the new Red value for coordinate 1,2. You must then repeat this for Green and Blue values. This must be repeated throughout the image. If you are working on a pixel which is not fully surrounded by pixels (8 pixels), you must take the average of however many neighbouring pixels there are.

Applying gaussian blur with 3x3 matrix using CUDA **(20 marks)**

Applying gaussian blur with multiple dimension matrices using CUDA **(extra 5 marks)**

NOTE: All coding marks are based on "academic judgement" of the work submitted. If your code compiles with minor errors, you will still gain marks based on the code however, many errors with no clear contribution will result in no marks.

Submission of work

Your completed work for assignments must be handed in on or before the due date. You must keep a copy or backup of any assessed work that you submit. Failure to do so may result in your having to repeat that piece of work.

Electronic submission

This is normally done via Canvas. Any special instructions will be available on the upload tag or within the assessment brief, (link will be provided).

Penalties for late submission of coursework

Standard University arrangements apply. ANY late submission will result in the grade 0 NS being allocated to the coursework.

Procedure for requesting extensions / mitigating circumstances

This is done via eVision. Further information can be found at http://www.wolvesunion.org/advice/academic/

Retrieval of Failure

Where a student fails a module (less than 40% for undergraduate modules, less than 50% for postgraduate modules) they have the right to attempt the failed assessment(s) once, at the next resit opportunity (normally July resit period). If a student fails assessment for a second time they have a right to repeat the module.

NOTE: Students who do not take their resit at the next available RESIT opportunity will be required to repeat the module.

Return of assignments

Assignments will be return during the module's feedback session If you have any questions regarding your feedback you normally have two working weeks from the date you receive your returned assessment and/or written feedback or receive your exam results to

contact and discuss the matter with your lecturer.

Cheating

Cheating is any attempt to gain unfair advantage by dishonest means and includes plagiarism and collusion. Cheating is a serious offence. You are advised to check the nature of each assessment. You must work individually unless it is a group assessment.

Cheating is defined as any attempt by a candidate to gain unfair advantage in an assessment by dishonest means, and includes e.g. all breaches of examination room rules, impersonating another candidate, falsifying data, and obtaining an examination paper in advance of its authorised release.

Plagiarism is defined as incorporating a significant amount of un-attributed direct quotation from, or un-attributed substantial paraphrasing of, the work of another.

Collusion occurs when two or more students collaborate to produce a piece of work to be submitted (in whole or part) for assessment and the work is presented as the work of one student alone. For further details see: http://www.wolvesunion.org/advice/academic/