MONASH University

FIT3143 Lab #2 (Week 3)

THREADS & OPENMP

OBJECTIVES

- The purpose of this lab is to introduce you to POSIX Threads and Open MP
- Design and develop parallel algorithms for various parallel computing architectures

MARKS

- The lab is worthed 5 of the unit final mark.

INSTRUCTIONS

1. Before class: Attempt all the tasks before class. Make sure you read the marking rubric and the lab pre-readings!
2. During class:
    - You will be given 15 minutes to prepare your code for the demonstration period.
    - When you have finished preparing the code, you are required to submit them to Moodle.
    - During the demonstration period, the teaching staff will ask you questions on your code submission(s).
3. Marks will not be awarded if you skip the class, do not make any submissions, or make an empty submission to Moodle (unless a special consideration extension has been approved).
4. You are allowed to use search engines or AI tools to search for information and resources during pre-class preparation. However, search engines and AI tools are not allowed during the code presentation period.
5. Always double test all your codes & written answers. Make sure the codes can be run on your laptop.

LAB ACTIVITIES

Prepare to answer the following questions.

Task 1 – Serial Code - Finding Prime Numbers (20%)

Write a serial C program to search for prime numbers which are strictly less than an integer n, provided by the user. The program should print to the standard output (screen) a list of all prime numbers found.

Example: For instance, if the user inputs n as 10 on the terminal, the prime numbers being printed on the screen are: 2, 3, 5, 7.

Hint:

- How to check if a number, let's say k, is prime? Yes, you can check if k is divisible by 2, by 3, by 4, by 5, …, by k-1.
- However, there is a (slightly) smarter way. Imagine k is not a prime number, there must exist two integers m and n such that m times n equals k. Will both m and n be larger than the square root of k? No! If m is larger than the square root of k, then n must be smaller than the square root of k, right?
- In other words, you don't need to check until k – 1 (think what happens when k is really large).
- For C: Make sure you know how to use the sqrt() of math.h and compile using the "-lm" flag of gcc.

Measure the time required to search for prime numbers less than an integer n when n = 1,000,000. Write down the answer in your notes and submit the serial code to Moodle in a file "task1.c".

Task 2 – POSIX Threads - Finding Prime Numbers (40%)

Write a parallel version of your serial code in C utilizing POSIX Threads.

In this part, you will need to design a parallel partitioning scheme distributing the workload among the threads, and implement your design in C.

After you implement the task, measure again the time required to search for prime numbers less than an integer n when n = 1,000,000 and compute the speed-up by your parallel implementation. Write down the speed-up answer in your notes. Make sure you also include information on the number of processors (cores) available in your machine and the number of threads you have created. In your notes, provide at least two reasons (with reference to your implementation) on why the speed up may not be the same as expected. Submit your code in this task to Moodle in a file "task2.c".

MONASH University

Task 3 – OpenMP - Finding Prime Numbers (40%)

Write a parallel version of your serial code in C utilizing OpenMP.

In this part, you will need to design a parallel partitioning scheme distributing the workload using OpenMP, and implement your design in C.

After you implement the task, measure again the time required to search for prime numbers less than an integer n when n = 1,000,000 and compute the speed-up by your OpenMP implementation. Write down the speed-up answer in your notes. Make sure you also include information on the number of processors (cores) available in your machine and the OpenMP configurations you have used. In your notes, provide at least two reasons (with reference to your implementation) on why the speed up may not be the same as expected. Submit your code in this task to Moodle in a file “task3.c”.

Reorganize your notes into a report and submit the report as either a Word document “report.docx” or PDF document “report.pdf”. Make sure all the answers are clearly marked and grouped according to the tasks.

Submission checklist:

- task1.c
- task2.c
- task3.c
- report.docx or report.pdf