Homework 1 Report

Concurrent programming, ID1217

Group members
[Sai Santhoshi Srinithya , darbha@kth.se]

Introduction

The aim of this assignment was to implement and analyze multi-threaded programs using the **pthread library in c**. The problem of critical section was also solved , condition variables were implemented and thread synchronization. The following two tasks were completed

1. Compute the Sum, Min, and Max of Matrix Elements

2. Compute pi

Task 1: Compute Sum, Min, and Max of Matrix Elements

Code implementation

The 3 programs are multi-threaded programs to compute the sum, minimum, and maximum of a matrix's elements along with the location(position or index) of the max and min element . The tasks were further split into three versions:

(a) Compute and display the sum, min, max, and their location(indices) concurrently using barrier.

(b) Ensure the main thread prints the results without using a barrier function or arrays for partial results.

(c) Use a "bag of tasks" approach to manage thread workloads dynamically.

1. In b and c shared variables are used to store the total value and the main thread will print the result once all the worker threads have terminated .

Results

The programs were executed and tested for matrix of size 1000 and below is the reported execution time with varying number of threads.

(It was also tested for other matrix sizes but execution times below are reported for a matrix of size 1000x1000.)

Number of threads	Execution time(ms)
2	0.987
4	0.973
6	0.745
8	0.758
10	0.830

Table 1: Execution time and no of threads- prgrm a

Number of threads	Execution time(ms)
2	1.113
4	1.033
6	0.805
8	0.804
10	0.921

Table 2: Execution time and no of threads- prgrm b

Number of threads	Execution time(ms)
2	1.118
4	0.937
6	0.888
8	0.925
10	1.009

Table 3: Execution time and no of threads- prgrm c

Task 2: Pi Computation Using Adaptive Quadrature

Code Implementation

This program computes π using adaptive quadrature by dividing the upper-right quadrant of the unit circle into sub-intervals. Each thread is responsible for calculating the area under the curve $f(x) = \sqrt{1 - x^2}$ within its assigned interval, recursively applying the quadrature method. The threads update the global π value, and a mutex is used to synchronize access to this shared resource. The approach ensures parallelism with the overhead of thread synchronization to compute π efficiently and accurately.

Results

The program was tested with different thread counts and ϵ values. The results for different thread sizes and epsilon value ($0.1 \times 10^{-8} \quad \text{or} \quad 1 \times 10^{-9}$) are presented in the table below:

Number of threads	**Execution time(ms)**
2	0.194
4	0.311
6	0.281
8	0.335
10	0.344

Table 4: Execution time for pi computation