Faculty of Engineering

Concurrent Programming — Project Proposal

Monte Carlo Pi Estimator

Submitted by:
Alaa Yehia — 6432
Ali Jaafar — 6411

Submitted to:
Dr. Mohammad Aoudi

Academic Year: 2024–2025
Date: 10/6/2025

1. **Introduction**

This proposal outlines our concurrent programming project, which aims to implement and evaluate a Monte Carlo Pi Estimator using both sequential and parallel approaches. Monte Carlo simulation is a probabilistic technique used to solve problems through random sampling. By applying this method to estimate the value of π, we aim to explore the benefits of parallel processing in Java.

Our objective is not only to demonstrate the accuracy of Monte Carlo methods in numerical estimation but also to showcase the performance gains achievable through multithreading and concurrent programming techniques.

2. Problem Statement

Estimating the value of π accurately is a classical computational problem. The challenge lies in optimizing this process to take advantage of modern multi-core processors. By comparing a single-threaded (sequential) implementation against a multi-threaded (parallel) one, we aim to assess the effectiveness of Java's concurrency mechanisms.

3. **Methodology**

The core algorithm randomly generates points within a unit square and determines whether each point lies inside the unit circle. The ratio of points inside the circle to the total number of points, multiplied by 4, estimates the value of π.

- **Sequential Approach:** A single-threaded loop using java.util.Random.
- **Parallel Approach:** Utilizes ExecutorService and Callable to divide tasks across multiple threads. Results are aggregated via a CompletionService.

4. Tools and Technologies

- Java Programming Language
- ExecutorService, Callable, Futures
- XChart for performance visualization
- GitHub for version control

5. Testing and Evaluation

The project is tested on sample sizes ranging from 10 million to 100 million points. Both execution time and estimation accuracy are recorded. We also compute the speed-up ratio to quantify performance improvements.

6. Expected Outcomes

- Demonstration of multithreading advantages in numerical simulations
- Cleaner and faster performance with increased sample size
- Visualization of performance comparison between sequential and parallel implementations

7. Future Enhancements

- Allow user to specify the number of threads
- Integrate higher-quality random generators (e.g., Sobol sequences)
- Expand estimator to other numerical integrations

8. References

GitHub Repository:
https://github.com/Alaa-F-Yehia/MonteCarloPiEstimator.git