

CS330- Introduction to Operating Systems - Course Project

Multithreaded Performance Optimization on Multi-Core Systems

Department of Computer Science, CCIS, PSU

Semester 252

Project Overview

Modern computing systems rely heavily on parallel processing to improve performance. Multi-core processors enable applications to perform tasks concurrently using multiple threads. Efficient multithreaded programming is essential in high-performance computing, data analytics, robotics, and real-time systems.

This project simulates a real-world performance engineering task where you will analyze, modify, and optimize a multithreaded Java program to evaluate performance improvements gained through parallel execution.

Realistic Scenario

You have joined a software engineering team developing a real-time data processing module for a smart agriculture analytics platform. The module processes thousands of sensor readings to compute derived metrics. Performance is critical because results must be delivered in real time.

The current system computes aggregate metrics using a single thread, causing delays. Your task is to evaluate and optimize the performance of the computation by implementing multithreading and measuring performance improvements on a multi-core processor.

Provided Files

Two Java files are provided that implement a multithreaded program for summing elements of an integer array. The program measures execution time using:

- Single thread
- Two threads
- Four threads

Example run-time results are included in the source file header.

Project Tasks

Part 1: Understanding Multithreading in Java

1. Study how multithreading is implemented in Java.
2. You may refer to:
 - Course lecture examples
 - Provided slides
 - Chapter 30 of Daniel Liang's Java Programming book
3. Import and run the program using NetBeans or any Java IDE.
4. Observe execution times for different thread counts.
5. Explain the observed performance differences.

Part 2: Performance Modification Task

Modify the program to compute:

Sum of cubes of array elements

Requirements:

- Limit array values to integers less than 100.
- Measure execution time for:
 - 1 thread
 - 2 threads
 - 4 threads
 - 6 threads
 - 8 threads
- Compute percentage improvement compared to the single-threaded version.

Performance Analysis

After collecting results:

- Compare execution times.
- Explain why performance improves or degrades.
- Discuss thread overhead, scheduling, and CPU core utilization.

Submission Requirements

Submit a zipped folder containing:

- Modified Java source code
- Results summary in tabular form (include multiple tables)
- Results table (see format below)
- Short explanation (maximum 1000 words)
- Graphs to show the performance trends
- Maximum threads after which the performance starts to decrease

Results Table Format

Threads	Execution Time (ms)	Speedup	% Improvement
1			
2			
4			
6			
8			

Important: Include machine specifications:

- Processor model
- Number of cores
- RAM
- Operating system

Learning Outcomes

By completing this project, students will:

- Understand multithreading and concurrency
- Analyze performance on multi-core processors
- Measure speedup and efficiency
- Understand thread overhead and scalability
- Apply performance optimization techniques

Marking Scheme (Total: 100 Marks)

Criteria	Marks
Understanding and explanation of provided code	15
Successful execution and verification of original program	10
Correct implementation of sum-of-squares modification	15
Implementation using multiple thread counts (1,2,4,6,8)	10
Accurate runtime measurement and data collection	10
Correct calculation of speedup and percentage improvement	10
Quality of performance analysis and explanation	15
Code quality, readability, and documentation	5
Proper results table and formatting	5
Machine specifications and reproducibility details	5
Total	100

Academic Integrity

All submissions must be your own work. Plagiarism or copied code will result in disciplinary action.

Submission Deadline

Submit via LMS before the announced deadline.