THREAD POOL DESIGN AND SIMULATION

Operating Systems Final Project - 4031 Semester

Navid P.Panahi

February 6, 2024

Contents

Introduction

In modern operating systems, efficient management of threads is crucial for optimizing resource utilization and improving application performance. One of the key techniques employed to achieve this is the use of thread pools. A **thread pool** is a software design pattern which maintains multiple threads waiting for tasks to be allocated for concurrent execution by the supervising program (further reading). In this project, You will design and implement a thread pool that allows multiple processes to request threads for executing tasks concurrently. The project also requires you to implement a simulation environment that will run the thread pool through various request sequences to test and validate its functionality. The environment must generate a detailed report at the end of the simulation.

Requirements

This section outlines the minimum requirements for the project, including the components necessary for each module.

Thread pool architecture

The thread pool must include the following components:

- Thread pool manager: This is the main component that manages the lifecycle of the thread pool. It is responsible for creating, managing, and destroying threads.
- Worker threads: These are the threads that perform the actual work. The thread pool manager creates a fixed number of worker threads that wait for tasks to execute.
- Task queue: A **thread-safe** queue that holds tasks waiting to be executed. The worker threads will pull tasks from this queue when they are available.

Simulation environment

The simulation environment should be designed to accept input parameters in a specified format that define the simulation settings. These parameters must include the following:

- The initial configuration of the thread pool (e.g. number of threads, size of task queue, ...)
- List of tasks or requests submitted to the thread pool including their arrival and execution times.

Once the parameters are provided, the environment should execute the simulation by submitting requests to the thread pool while logging relevant events and information, such as thread allocation times and task completion times. Upon completion of the simulation, the environment should generate a comprehensive report that includes the collected logs and final results, such as the total simulation time, the number of thread allocations, and the number of unused threads.

Implementation Details

Consider the following points as you work on the project:

- Unlike real operating systems, where each task may require several threads to fully execute, each task requires only a single thread to execute.
- Implement locks, mutexes, and other concurrency management techniques to ensure the thread safety of the task queue.
- Utilize multi-threaded programming techniques to enable the thread pool to manage multiple requests concurrently.
- Minimize the use of libraries and packages. Excessive or unnecessary usage will incur penalties.
- Feel free to make assumptions regarding any unspecified details of the project.

Deliverables

Your submission must include the following deliverables:

- Source code
- Three input files along with their corresponding final reports generated by the simulation environment
- A README file that explains how to run the program