# Lab 1: C++ Threading

Multithreaded PROG/ARCH/TOOLS, Fall 2025

Name: Yu-Ting Cheng

EID: yc35637

For this lab, I chose the project “Parallel Processing Pipeline with Semaphores”. I use a 3 stages pipeline structure to implement genetic algorithm.

## Overview

Genetic algorithm consists of parent individual selection, crossover operation and cost function calculation. In this project, I define the 3 stages pipeline with the following structure:

- Selection: Select two parent individuals from parents' pool, then push them to the bounded queue.
- Crossover: Pop two individuals from the queue, perform crossover operation and produce two offsprings. Finally, push them to the bounded queue for next stage.
- Cost: Pop two individuals from the queue, calculate their cost function value, and then add them to the offspring pool.

Initially, there are 100 parents in the parents' pool. User can specify the number of offsprings they want to produce and the number of threads in each stage. To run the code, please refer to “How to run?” section in README.md

## Experiment Results

The following table shows the latency of different number of offsprings and number of threads. The following data are measure multiple times and take average from them. For simplicity, I use the same number of threads in each stage.

| Number of Offsprings | 1 thread/stage | 2 threads/stage | 4 threads/stage | 8 threads/stage |
|---|---|---|---|---|
| 16 | 112.6 ms | 96.2 ms | 83.8 ms | 76.4 ms |
| 64 | 188.5 ms | 183.2 ms | 104 ms | 100 ms |

In addition to latency, I also recorded the value of cost in offspring pool. Since I always select the parents with highest score in selection stage, I expect to get offsprings with higher score according to genetic algorithm. In my experiment, I observe the offspring pool has an approximate **25%** increase compared to parents' pool, which verified the assumption of genetic algorithm.

## Challenges

This is my first time to implement a multithread program, which is quite different from sequential programming in most C++ program. The most common challenge I faced with is debugging. Since the program is running multithreaded, it's hard to find the bug through traditional "cout" command, as threads are executing concurrently.

In addition, I got confused about how many operations each should execute for a while. Initially, I use a while loop and a global variable (e.g., selection_todo) to track the number of tasks remaining in this stage. However, I found that while loop is dangerous in exiting function. On the other hand, using global variable may potentially result in **false sharing** across multithreads. Therefore, I ended up no using global variable and use a for loop to execute tasks for each thread in stage.

## AI Tool

I use UT Spark to generate testcase and test plan.