Computação em Larga Escala

(ano letivo 2024'25)

Practical Assignment 1

Transitioning from Single-Threaded to Multi-Threaded Applications Using C++ `std::thread`

Objective

The goal of this assignment is convert the **`weather-stations`** and **`word-count`** exercices, for which you have developed single-threaded solutions, into multi-threaded applications running on Linux/Unix.

Your solution should be implemented in C++ using `std::thread` from the C++ Standard Library to create and manage threads. Ensure proper **synchronization** where necessary (e.g., using **mutexes** or **condition variables**) to avoid race conditions.

Requirements

1. You must create a thread pool with a fixed number of workers. The pool should allow enqueuing an arbitrary number of tasks and provide the capability to wait until all tasks have been completed. The number of workers is an input argument.
2. Groups of three members are required to implement a multi-threaded solution to sort the output of the weather-stations result based on the maximum temperature.
3. Ensure that your implementation is properly validated:
    - **Verify correctness**: Confirm that the multi-threaded implementation produces results identical to the single-threaded version.
    - **Compare execution time**: Measure and compare the runtime of both the single-threaded and multi-threaded implementations.
    - **Evaluate performance improvements**: Calculate speedup and efficiency to assess the benefits of parallelization.

Grading

- Development and validation of a multithreaded version of the `weather-stations` problem according to specification – 13 points
- Development and validation of a multithreaded version of the `weather-stations` and `word-count` problems according to specification – 20 points

Deliverable

You should host your source code repository on **GitHub**. Ensure that your repository includes:

- **All source files** for both single-threaded and multi-threaded implementations.
- A **README.md** file with setup instructions, usage details, and a summary of your approach.
- **Performance analysis** results, including execution time comparisons and speedup calculations.

Deadline

March 19, at midnight.