Ömer Faruk Ünal
2019400048

Threading, or not?

Usage

- To run the project simply write `make run n={Length of Array}(make run n=1000).` This will create three output files with names “output1.txt” “output2.txt” “output3.txt”
- Output1 is the result of oneThread, output2 is the result of fiveThreads and output3 is the result of tenThreads.
- To create an executable file just write `make`. This will create three executable files with name “oneThread” “fiveThreads” “tenThreads”
- To clean the executables and .txt files run `make clean.`

Description

- This program takes integer n as an input. And creates a random array with length of n.
- Then calculates •Min • Max • Range • Mode • Median • Sum • Arithmetic Mean • Harmonic Mean • Standard Deviation • Interquartile Range of the array.
- If there are two modes, returns the mod value with the smallest index in the original list.
- Double precision may cause some minor mistakes in larger numbers.

Project Outline and Explanations

This project was written in the C++ language, and it consists of three `.cpp` files. These files are:

- `oneThread.cpp`: There is just main thread, and all tasks are assigned to main thread.
- `fiveThreads.cpp`: Five threads are created, and each thread has two tasks to finish.
- `tenThreads.cpp`: Ten threads are created, and each thread has one task to finish.

In one thread version array is firstly sorted and then calculations have been made. But in other codes since we just need to sort the array to find median and IQR, the threads assigned to those functions sort the array. This way we prevent unnecessary time lost while sorting the array.

In five threads version I split the tasks as minMax, medianIqr, modeRange, sumMean, stdHmean to prevent unnecessary calculations. After finding sum I can find mean in O(1) so it is assigned to one thread to find both sum and mean.

In ten threads version each task is assigned to one thread. There are some unnecessary calculations in this approach. For example, in this approach finding min, max and range is in different threads. However, we can find all of these with just one loop. So, creating threads have and overhead.

For n smaller than 10000, the single threaded version is almost twice as fast as the others. Because we waste time creating, joining new threads and assigning them to CPU, and the time we lose is more than the time we gain.

For larger n values, threaded versions become faster because this time, calculations are large and this saves us time when we do more than one calculation at the same time. However, the 10 Threaded and 5 Threaded versions work almost at the same speed. The reason for this is that we have to do some unnecessary calculations and list sorting over and over as I mentioned above.

So the conclusion we can draw from here is that although Threads allow us to perform multiple calculations at the same time, sometimes the additional loads they bring can exceed our earnings. That's why it's not good to create threads for every calculation.