Lab 9: Code Profiling

November 7, 2020

Introduction

This lab will give you experience using both CPU profilers. Please make an answers file for both problems and commit it to the repository along with your code fixes.

Problem 1: Callgrind

This problem's code appends one random number between 1 and 1000 to a vector and prints the average of the vector. Hopefully, the average will converge to around 500. Include your answers in `answers.txt`.

1. Look at the profile for the methods defined in `vector.h` . What are the two most called methods, and how many times are they called?
2. Look at Vector's copy constructor. How many times is it being called?
3. Why is it being called that many times?
4. Fix the code to not make unnecessary copies.
5. Run the code through callgrind again. Did your fix work? How many times is Vector's copy constructor being called now?

Problem 2:

This problem's code calculates the length of input lines and prints a running average. Build `prob2.cpp` and run it through callgrind (it needs input). Include your answers in `answers.txt`.

1. What line in `main()` consumes the most instructions?
2. How many *instruction reads* (Ir) are spent calculating the average? (lines 17-27)
3. Look at the source code. Do we really need to re-count the length of each line each time we calculate the average? (Hint: no.) Fix the code to maintain a running total of characters and a line count. (This is not a one-line change; please throw away a lot of this slow implementation.)
4. Re-build and run the code through `callgrind` . How many instructions are spent calculating the average now?

Epilogue

Make sure to add and commit your makefile, gitignore, and fixes to the various problems and push your commits to GitLab!