Assignment P3 – Not a Drill

Formal assignment description - INFOMOV
Jacco Bikker – May - June 2024

Introduction

This document describes the requirements for the final assignment for the INFOMOV course. For this assignment you will optimize an application of your own choice, by applying the structured optimization process, using the means discussed in the course.

Application

For this assignment you may use any application as a starting point. The application may be your own work, or an open source project (such as llama.cpp), or the compact 'safety net' project provided via the INFOMOV website.

The project you optimize may be written in C++ or any other language. It may be near-optimal or not optimal at all. It may run on Windows, Linux, or something exotic. Do however pick a project that allows you to showcase your skills. Do not optimize by just porting from a 'slow' language (e.g. Java, C#) to a faster one (e.g. C++, Rust).

Assignment Goal

For this assignment, all options are on the table. You may optimize using SIMD, multithreading, GPGPU, DOD and any other technique presented in the lectures or found elsewhere. The goal is to make an application faster.

That being said: You are expected to apply optimizations using a **structured approach**, and to **document the process**. In fact, your report will be your primary product for this assignment.

Make sure you clearly define what the optimization goal is for your chosen application. Make sure the chosen project **lets you assess performance** in detail, both at a global scope, and at a finer level (probably using a profiler). This may require modifications to the software to achieve **deterministic** and **reproduceable runs** without human input.

Structured Process

Recall the structured approach to program optimization:

0. Determine the optimization goals.
1. Profile to determine bottlenecks.
2. Apply high-level optimizations (reduce algorithmic complexity) to the bottlenecks.
3. Profile again.
4. Apply vectorization, GPGPU, and/or multi-threading, if applicable.
5. Profile.
6. Apply low-level optimizations.
7. Repeat steps 5 and 6 until time runs out.
8. Report.

A successful assignment 3 submission follows and documents these steps in detail, with concise profiling results and an overview of the steps that lead to a faster application. Note that you do not have to document failed attempts.

Depending on the project you may or may not actually apply steps. Profiling should however always play a central role, and optimization effort should be aimed at the determined bottlenecks. Make sure your report closely documents how you approach this and what decisions you made.

Don't limit yourself to 'just' high-level optimizations; show your skills in the other steps as well!

Team

As usual, you may work on this assignment alone, or with one partner. You may team with one partner for all assignments, but it is also allowed to change teams per assignment. You cannot change your team halfway an assignment; if for whatever reason you don't want to finish the project with your partner, both of you will work alone. Both team members may continue working with the code that was produced up till the split.

You may exchange information about the project with other students, online or in real life. Do not share code snippets, limit the exchange to ideas, hints, concepts and taunts.

Grading

Your final grade will be determined **primarily by your report**, which should describe how you approached the optimization. Grades will be assigned roughly as follows:

- 4 and lower: your application does not work, did not improve or has broken functionality after the optimization.
- 5: you did make the application faster (perhaps even significantly so), but it is not clear why, or the report does not contain proof that the optimization was conducted in a structured manner.
- 6: you did make the application faster, but you ignored bottlenecks indicated by the profiler, because you got distracted or perhaps because you didn't master obviously applicable technology, such as SIMD or GPGPU.
- 7: this is the baseline: reasonable speedup, approach and report.
- 8: above average: ~90% of the potential was realized, thanks to diligent profiling and a clearly documented approach.
- 9: exceptional: close to 100% of the potential was realized, profiling now ping-pongs between a few already optimized bottlenecks. Or: 8, plus acceptance by the open source community of your improvements ("peer-reviewed optimizations").

The following grading scheme will be used, with each item on a scale of 1..5:

15% - "Project Complexity", 1=200 lines of code, 5=optimization involves 50k lines of code or more.
20% - "Speed increase", 1=no improvement, 4=fully realized, 5=beyond expectation.
40% - "Structured Process", 1=unclear or undocumented, 5=amazingly structured report.
15% - "Profiling Accuracy", 1=noisy/imprecise/inconsistent, 5=accurate, concise, relevant.
10% - "Bonus": 0, 0.5 or 1 point, for something unforeseen or special.

Deliverables

Your submission should consist of the optimized code plus project files. If any tools beyond Visual Studio are required to produce the intended executable, please add a `readme` that contains instructions.

Apart from the project you are expected to produce a report. This report is in most cases your primary deliverable.

Deadline

The deadline for this assignment is **Wednesday, June 19th, 17:00**. As usual you may submit your work up to 24 hours later; the penalty for this is 1 point.

Academic Conduct

The work you hand in must be your own original work, or properly referenced. If you used materials from other sources, please specify this clearly in the `readme.txt`.

Random Links

Perhaps you will find some inspiration on the following links, mostly unchecked:

- https://github.com/fffaraz/awesome-cpp
- https://github.com/jhonnold/berserk
- https://github.com/fogleman/Craft
- https://sdcc.sourceforge.net/

The End

May The Light be With You! - Questions and comments:

bikker.j@gmail.com

INFOMOV 2024