Concurrent & Parallel Systems - Coursework 1 Report

Glenn Wilkie-Sullivan - 40208762

October 12, 2018

Abstract

This report will detail a project attempting to add concurrency and parallelism within a block-chain simulator in C++. The project in question will utilise methods such as multi-threading, algorithmic skeletons, CPU-level parallelism and OpenMP (Open Multi-Processing) to optimise the simulator as much as possible.

1 Introduction and Background

2 Initial Analysis

To analyse the performance and overall CPU usage of the blockchain application, I used the Visual Studio 2017 diagnostic tools. To start with, I ran the application to examine the overall performance, as well as any potential bottlenecks in the code. Unsurprisingly, the performance of the application hinges mainly on the 'difficulty' variable, which the programmer can control. After 5 runs, the application, with difficulty 5, mined 5 blocks in about 43 $\pm$ 10 seconds. With difficulty 6, the application mined 5 blocks in about 382 $\pm$ 30 seconds.
As for bottlenecking, the application has a few instances of it. The most effective method of finding the bottleneck was to use Visual Studio's debugging tools, between a range of difficulties. The results after running the application with difficulty 6 was as follows:

3 Methodology

4 Results and Discussion

5 Conclusion

6 References

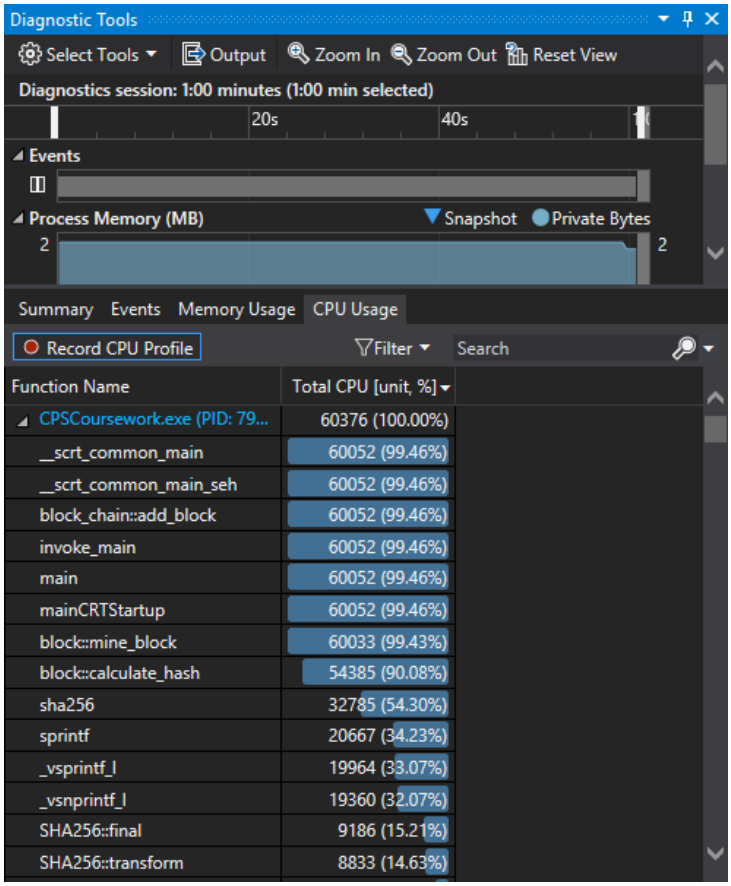

Figure 1: Difficulty 6 CPU Usage