ECE 459: Programming for Performance
Assignment 4

Sean Byungyoon Kim

April 5, 2019

Option 1: Profiling

To profile this assignment, I used the profiler provided by Linux called perf. With this profiler, I created a flame graph that outlines the performance of the original program before any changes were made to the code. This flame graph can be found in the included folder (flame) titled: "perf_fast_original.svg".

From the original flame graph, it can be seen right at the start that some obvious changes can be made. The first change that was made was to change the implementation of the Container class because it's push and pop operations were very expensive and took a lot of time. Instead of an implementation involving arrays, I changed the data structure to a linked list due to its fast operations for inserting and deleting nodes. As well, using a linked list eliminated the requirement or resizing the container to accommodate larger data sizes.

Another change I made to the program was changing when the program reads data from the .txt files. Previously, the data from "ideas-products.txt" and "idea-customers.txt" were read in every time getNextIdea was run when each IdeaGenerator was ran. Instead, the files are only read at the very start when IdeaGenerator::run is called to minimize the number of times data has to be read from the .txt files.

A similar change was made within PackageDownloader.cpp, where previously the data from "packages.txt" was being read in every iteration of the for loop within PackageDownloader::run. Instead, the file is now read at the start of PackageDownloader::run and the ifstream object is passed into the readFileLine function to minimize the number of times the ifstream object has to be created from the .txt file.

The final changes involved changing the how the program calculates its xorCheckSum. Previously, the program utilized stringstreams and strings from the std library. However, operations involving those variables are slow, so the creation of bytes for the xorCheckSum were changed to use bitwise operations for their low performance overhead. The final flamegraph can be found in the included folder titled: "perf_fast_final.svg". The performance benchmarks can be found in the table below as well. The program was run on ecetesla0 using the command "time ./hackathon_fast".

	Time (s)
Run 1	0.048
Run 2	0.056
Run 3	0.046
Run 4	0.049
Run 5	0.051
Average	0.050

Table 1: Benchmark results for hackathon_fast on ecetesla0