Computing Science and Mathematics
University of Stirling

Accelerating Financial Calculations Using Haskell and the GPU

Alex Olivier Appetiti

Submitted in partial fulfilment of the requirements for the degree of
MSc. in Computing for Financial Markets

September 2014

Abstract

A simplified financial simulation often run by Prudential is translated from idiomatic multicore C code into GPU (Graphics Processing Unit) code via Accelerate - a DSL (Domain Specific Language) for the Haskell programming language - to show that GPUs can be used to speed up heavy financial computations without sacrificing code legibility, maintainability and correctness.

Many problems tackled within the field of computational finance involve aggressive 'brute-force' simulations which consist in running the same function(s) on a large number of different inputs. This is often a time consuming process which forces companies to make compromises by limiting the range of the simulated inputs or sacrificing the precision of the simulation. These incomplete simulations are often undesirable as they can lead to less profitable decision-making.

Furthermore, the effort to expedite the execution of the financial evaluations often pushes software developers to produce very low level code, which is notoriously harder to maintain as well as being more bug-prone. Not only does this imply that companies run the risk of relying on broken simulations, but it also means that more effort and resources need to be channeled into maintenance and debugging rather than solving actual problems.

By compiling *high-level, strongly-typed, functional* Haskell code onto the *GPU* it is possible kill two birds with one stone; the highly parallel nature of running many nearly identical simulations make such a problem ideal for high-speed execution on a GPU, without necessarily giving up the legibility or correctness of the code.

Two micro-benchmarks and a commonly used financial model (known as the IL Model) are implemented in both idiomatic multicore C, Haskell and GPU-Accelerated Haskell. For each benchmark, both the legibility of the final program and the execution speed are discussed to evaluate whether or not the newly proposed solution is worth exploring. In the final benchmark, several program-transformation methods are shown in order to allow for the paradigm shift between regular CPU computing and GPU computing.

The final experimental results show that given the hardware used, the GPU code manages to execute the proposed simulation faster than optimised parallel C code, showing great promise for its potential.