Exploring Parallel Programming in Haskell

Rashad Gover

May 11, 2021

Contents

Abstract

In this report, we will explore parallel computation in the pure functional language ***Haskell***.

1 Introduction to Haskell

This is the introduction to Haskell, its' properties, and syntax.

1.1 Syntax

This section goes over the syntax of Haskell

1.2 Fibonacci Sequence

This section will go over the code in Haskell for the Fibonacci Sequence

1.3 Matrix Multiplication

This section will go over the code in Haskell for Matrix Multiplication

1.4 Benchmarking

This section goes over Threadscope and benchmarks the two examples above. Fibonacci and Matrix Mutiplication.

2 Parallel Evaluation Strategies

This section will breifly explain Strategies and the Eval monad, and the functions rpar and rseq.

2.1 Results

This section will have the results for the Fibonacci Sequence and Matrix Multiplication using Strategies.

3 Dataflow Parallelism with monad-par

This section will introduce the Par Monad and the concept of dataflow parallelism.

3.1 Results

This section will have the results for the Fibonacci Sequence and Matrix Multiplication usingthe Par Monad.

4 Accelerate Library

This section goes over the Accerlerate library and the idea of embedded DSLs withing Haskell, for which it is known for. In this section, HW2 particle simulation is used

4.1 Particle Simulation

This section describes the Particle Simulation algorithm in some detail and HW2.

4.2 Multicore Programming with Accelerate

This section goes over multicore programming in Accelerate.

4.2.1 Results & Comparison with OpenMP

In this section we compare the results with the OpenMP variant of the HW.

4.3 GPU Programming with Accelerate

This section goes over GPU programming in Accelerate. This will be pretty much the same as the multicore programming in the above section because we can write programs for both platforms with a single syntax! The syntax used to describe the parallel program is decoupled from the platform it runs on which is great.

4.3.1 Results & Comparison with CUDA

This section will go over the benchmark results and compare it with the CUDA implementation of HW2.

5 Conclusion

This section will rap up the results of the benchmarks and come to conclusion. Was climbing up the abstraction ladder worth the tradeoff in performance?

Was the drop in performance of the Haskell implementations worth the better ergonomics? This could be argued, especially if the user isn't familiar with functional programming. Was the performance of the Haskell programs not that much of a difference from the C++ implementations? What do overhead costs look like? How does scaling compare between the two platforms? Weak and strong scaling.