Lab D Parallel Functional Programming

Part 1

The first task is a variant on the stock market problem, which is a classic NESL example. The input is an array of **Int**s, corresponding to the price of a stock on successive days. You may, if you wish, assume that the input has length a power of two. A trader wants to make either zero or one transactions consisting of buying on one day and selling on another, later day. Write a program using the **Repa** library that, given such an array of prices, returns a triple **(b,s,p)**, where **b** and **s** indicate the indices into the array at which it is best to buy and sell the stock respectively, and **p** is the resulting profit (the difference between the prices at the two times). If two different times for sale give the same profit, then choose the earlier one, and similarly if two times for buying give the same profit, choose the later one. Return **(0,0,0.0)** or in some other way indicate when no transaction is wise.

Example:

***Main> let ins = fromList (Z :. (8::Int)) [0,0,2,9,8,10,1,10] :: Array U DIM1 Int**

Applying your **buySell** function to **ins** should give **(1,5,10)**

Benchmark using Criterion and report on performance and speedups (if any). The aim is to get you to use **Repa**, and at the same time to start thinking about cost models in NESL. Don't worry if performance is disappointing.

Analyse the cost of your algorithm in terms of work and depth (or span) in the style of Blelloch, for an input array of length **N**. I would like you to devote significant time to this part of the lab, studying relevant material (including the lecture on Data Parallel Programming, which introduces NESL, and the associated reading). The best place to start is probably on the page about NESL at CMU. You can then move on to papers by Blelloch.

Part 2

You are required to do EITHER Task I or Task II.

(You can do both if you wish.)

Task I: write a tutorial on one of the following topics:

Repa 3: a tutorial for curious Haskell programmers

Data Parallel Haskell: a tutorial for curious Haskell programmers

Profiling and optimising parallel Haskell programs with Threadscope: a tutorial

The Par Monad for parallel Haskell programming: a tutorial

A comparison of different approaches to deterministic parallel programming in Haskell

A tutorial on Parallel Strategies in Haskell

How to use `par` and `pseq` for parallel programming in Haskell

Why choose Haskell for deterministic parallel programming?

Single Assignment C: a tutorial

Parallel functional programming in Java 8: a tutorial

GPU programming in Haskell: a tutorial on Accelerate

Parallel sorting in Haskell: how well can we do?

An introduction to skeletons and their use in parallel functional programming

Parallel programming in F#: a tutorial

Laziness considered harmful for parallel programming

or on a topic of your choice agreed with Mary by email (ms at chalmers.se). Erlang related topics are welcome. This year, we will (as an experiment) not try to restrict each topic to being covered by one lab group.

The main point, remember, is to make a clear and simple tutorial (something like a technical blog post). Good tutorials often contain nice pictures! Give readers some information about sequential and parallel running times. Submit your document or web page with all associated images and code. We may add excellent tutorials to the course pages.

Task II: Implement an interesting deterministic parallel program in Haskell, documenting your work in developing and optimising the program, and writing a blog post about it. We will be particularly impressed if you in some way augment the parallel library that you are using. If you are in doubt about your choice of problem to tackle, consult with Markus and Anton, copying to Mary. You should put in about the same amount of work as you would in writing a tutorial (Task I above). This task might be a good way to start exploring possible Masters thesis topics.