Harnessing the Multicores: Nested Data Parallelism in Haskell

Simon Peyton Jones

Microsoft Research, Cambridge, U.K

Abstract. If you want to program a parallel computer, a purely functional language like Haskell is a promising starting point. Since the language is pure, it is by-default safe for parallel evaluation, whereas imperative languages are by-default unsafe. But that doesn't make it easy! Indeed it has proved quite difficult to get robust, scalable performance increases through parallel functional programming, especially as the number of processors increases.

A particularly promising and well-studied approach to employing large numbers of processors is to use data parallelism. Blelloch's pioneering work on NESL showed that it was possible to combine a rather flexible programming model (nested data parallelism) with a fast, scalable execution model (flat data parallelism). In this talk I will describe Data Parallel Haskell, which embodies nested data parallelism in a modern, general-purpose language, implemented in a state-of-the-art compiler, GHC. I will focus particularly on the vectorisation transformation, which transforms nested to flat data parallelism, and I hope to present performance numbers.

G. Ramalingam (Ed.): APLAS 2008, LNCS 5356, p. 138, 2008.