発表概要

Glasgow Haskell Compilerを用いたプロファイル駆動型自動並列化機構の簡潔実装

新井 淳也[1,a)]　前田 俊行[1]　石川 裕[1]　米澤 明憲[2]

2011年7月29日発表

本発表ではGlasgow Haskell Compiler（GHC）をベースとしたHaskell用自動並列化コンパイラの簡潔な実装を提案する．近年，マルチコアプロセッサの普及により並列プログラミングの重要性が増している．Haskellは原則として式に副作用がないため高い並列性を持ち，Haskellの並列化に関しては多くの研究が存在する．過去の研究は並列プログラミングを簡単にするための様々な仕組みを提供してきたが，それらのほとんどは並列実行に適したコードブロックの発見を人の手で行う必要があった．並列化は実行時のオーバヘッドを招くため，並列化においては並列性があるだけでなく性能向上に貢献するコードブロックを発見することが重要となる．そこで本研究では，プログラムのプロファイリングおよびプロファイルの分析による並列化に適したコードブロックの発見を自動化し，その仕組みを自動並列化のために使用した．自動並列化に関する先行研究は存在するが，それらにおける並列化の実装はコンパイラや実行時ライブラリに対する相当な変更を必要としたのに対し，本手法はGHCに組み込みのプロファイラによって得られる計測結果を活用できるよう設計されているため，GHCに対し小さな変更しか必要としないという意味でより簡潔である．本手法の効果は，いくつかのベンチマークプログラムをコンパイルし並列化することによって検証した．

Lightweight Implementation for Profile-driven Implicit Parallelization on the Glasgow Haskell Compiler

Junya Arai[1,a)]　Toshiyuki Maeda[1]　Yutaka Ishikawa[1]
Akinori Yonezawa[2]

Presented: July 29, 2011

This presentation proposes lightweight implementation of the Haskell compiler based on Glasgow Haskell Compiler (GHC) to perform automatic parallelization. In recent years, multi-core processors have prevailed and importance of a parallel programming is growing. Haskell has a lot of potential for parallelism because expressions are basically side-effect free, and many studies about parallelization of Haskell have been conducted. The previous studies have provided various mechanisms for painless parallel programming; however, most of them still require manual work to find proper code blocks to be executed parallel. Since parallelization incurs runtime overheads, it is essential to find code blocks that are not only parallelizable but also contribute to improving performance. Therefore, we automated detection of the code blocks suitable for parallelization through profiling a program and analyzing the result, and applied the mechanism to implicit parallelization. There have been precedent research about implicit parallelization, but their implementation for parallelization requires substantial modification to compilers or runtime libraries. On the other hand, the proposed approach is lightweight in the sense that it requires small modification to GHC because it is designed to be possible to utilize the result of the profiler embedded in GHC. We investigated effectiveness of this approach through compiling and parallelizing several benchmark programs.

[1] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[2] 計算科学研究機構
Advanced Institute for Computational Science, Kobe, Hyogo 650–0047, Japan
[a)] arai@is.s.u-tokyo.ac.jp