7H-02

OSCARベクトルマルチコアアーキテクチャのコンパイルフロー構築及び評価

高橋　健 *　　狩野　哲史 *　　宮本　一輝 *　　河田　巧 *　　柏俣　智哉 *　　牧田　哲也 *
北村　俊明 †　　木村　啓二 *　　笠原　博徳 *
* 早稲田大学理工学術院情報理工学科　　† 早稲田大学アドバンスドマルチコア研究所

1　はじめに

科学技術計算や画像処理、機械学習などの大量の演算処理を要するアプリケーションでGPUなどのアクセラレータが利用されている。アクセラレータ用のプログラミングには、GPUにおけるCUDAやOpenCL等の特殊なAPIや独自拡張のプログラム言語の利用や、場合によってはアセンブリなどのより低位の環境による開発が必要になる。さらに、アプリケーションの特性とアクセラレータのアーキテクチャに対する深い理解、及び特殊なコーディング技法が必要となり、これらがアクセラレータの性能を引き出すプログラム開発を困難にしている。

一方、筆者等は、OSCARコンパイラ[1]による自動並列化と、OSCARベクトルマルチコアアーキテクチャの開発によってこれらの問題を解決しようとしている。 OSCARコンパイラは、逐次のC言語で書かれたプログラムに対して並列化や低消費電力化などの最適化を適用し、OSCAR API[2]を用いた並列化Cプログラムを生成する。また、OSCARベクトルマルチコアアーキテクチャは、OSCARコンパイラとの協調により高性能・低消費電力のシステムを実現するためのマルチコアアーキテクチャである。本マルチコアアーキテクチャでは、従来からのコンパイラによる最適化技術の蓄積があるベクトルアクセラレータと、これに対してデータを供給するローカルメモリ及びデータ転送ユニットを各コアが持つ。これらにより、ソフトウェア開発者はアクセラレータのための記述を行わずにアクセラレータを利用可能となる。

本稿では、OSCARコンパイラによりCプログラムを自動並列化及びベクトル化し、OSCARベクトルマルチコアアーキテクチャのFPGAエミュレータ実行バイナリを出力するコンパイルフローを提案する。さらに、行列積及びNAS Parallel BenchmerkをOSCARコンパイラでコンパイルしFPGAエミュレータで評価した結果を報告する。

Development of Compilation Flow and Evaluation of OSCAR Vector Multicore Architecture
*Dept. Computer Sci. Eng., Waseda Univ.
†Advanced Multicore Processor Research Institute, Waseda Univ.

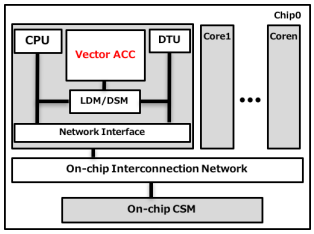

図1　OSCARベクトルマルチコアアーキテクチャ

2　OSCARコンパイラ

OSCARコンパイラはC及びFortranを対象とする自動並列化コンパイラである。まず、入力であるC及びFortranの逐次ソースコードに対してプログラム全域にわたり並列性を解析し、その結果としてマクロタスクグラフとして生成する。その後、リストラクチャリングとマルチグレイン並列化・ベクトル化・メモリ管理・ソフトウェアキャッシュコヒーレンシ制御・低消費電力化等の最適化を行う。上記の最適化後、OSCAR APIを用いた並列化ソースコードを出力する。

3　OSCARベクトル マルチコアアーキテクチャ

OSCARベクトルマルチコアアーキテクチャは、現在開発中のアーキテクチャであり、その評価環境としてソフトウェアシミュレータとFPGAエミュレータに実装が進められている。このアーキテクチャはCPU・ベクトルアクセラレータ・データ転送ユニットを持つプロセッサエレメント(PE)を複数持ち、各PEはベクトルアクセラレータにデータを共有するローカルメモリ（ローカルデータメモリ: LDM、分散共有メモリ: DSM）を持つ。また、各PEには集中共有メモリ（CSM）が接続される。 OSCARベクトルマルチコアアーキテクチャ図を図1に示す。

4　コンパイルフロー

提案するOSCARベクトルマルチコア用コンパイルフローを以下に述べる。

まず、逐次のCソースコードを入力とし、

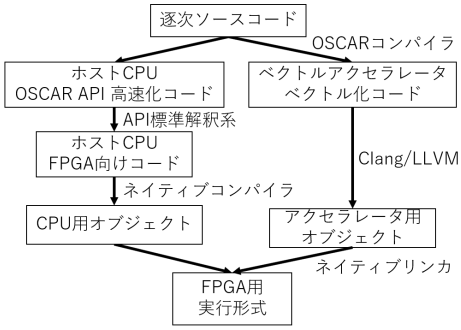

図2　OSCARベクトルマルチコアのコンパイルフロー

OSCARコンパイラを用いて並列化、ベクトル化等の高速化を施し、ホストCPUで実行するOSCAR API付きソースコードとベクトルアクセラレータで実行するソースコードを分けて出力する。次に、ホストCPU用のOSCAR API付きソースコードを入力とし、OSCAR API標準解釈系[3]を用いて、OSCAR APIをターゲットコンパイラが解釈可能なソースコードへ変換し、FPGAエミュレータ用のCソースコードを出力する。このFPGAエミュレータ用のCソースコードを入力とし、ネイティブコンパイラを用いて、ホストCPU用オブジェクトを出力する。また、ベクトルアクセラレータ用のソースコードを入力とし、Clang/LLVMを用いて、アクセラレータ用オブジェクトを出力する。最後に、ホストCPU用オブジェクト、アクセラレータ用オブジェクト、各ライブラリを入力とし、ネイティブリンカを用いて、FPGAエミュレータ用並列実行バイナリを出力する。本コンパイルフローを図2に示す。

5　評価

本節では、提案するコンパイルフローを用いて、行列積及びNAS Parallel Benchmarkを自動並列化及びベクトル化し、OSCARベクトルマルチコアアーキテクチャのFPGAエミュレータで評価した結果を述べる。

本評価では、OSCARベクトルマルチコアアーキテクチャをFPGA上に再現したエミュレータを用いた。CPUコアにはNIOSコアを用いた。本エミュレータの実装として、ベクトルアクセラレータを持たない4コア構成のエミュレータをDE-Nano Kit上に構築したもの（実装1）と、1コア+ベクトルアクセラレータ構成をArria10 SoC Development kit上に構築したもの（実装2）を用意した。

実装1の周波数は50MHz、データキャッシュ・命令キャッシュ共に32KBであり、ハードウェアでのキャッシュコヒーレンシ制御はなく、OSCARコンパイラによるソフトウェアコヒーレンシ制御を適用した。

実装2の周波数は100MHz、データキャッシュ・命令キャッシュ共に64KBである。

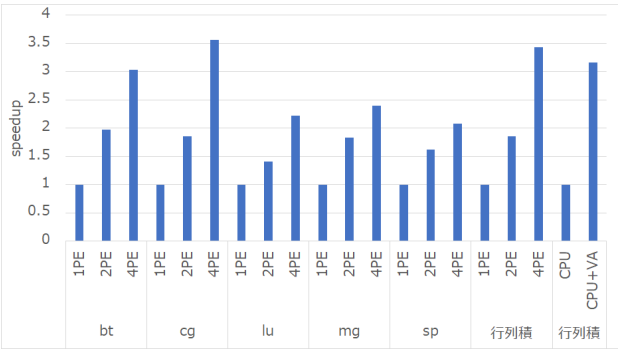

図3　FPGAエミュレータ上での評価結果

本評価では、評価アプリケーションに100x100の行列積及びNAS Parallel BenchmarkのBT、CG、LU、MG、SPのCLASS Sを用いた。

評価アプリケーションを構築したコンパイルフローを用いて自動並列化及びベクトル化し、FPGAエミュレータでエミュレートしたときの速度向上率を図3に示す。マルチコアによる並列実行によりスケーラブルな性能向上が得られ、4PEで平均2.78倍、最大3.60倍の速度向上を得られた。また、ベクトル化により1コアに対して3.17倍の速度向上が得られた。

6　まとめ

本稿では、OSCARコンパイラを用いて逐次のCソースコードを自動並列化及びベクトル化し、OSCARベクトルマルチコアアーキテクチャのFPGAエミュレータ用実行形式を出力するまでのコンパイルフローを提案した。また、構築したコンパイルフローを用いて、行列積及びNAS Parallel Benchmarkを自動並列化しFPGAエミュレータ上で実行させた結果、各アプリケーションで並列実行により4PEで平均2.78倍、最大3.60倍の速度向上を得られた。また、ベクトル化により1コアに対して3.17倍の速度向上が得られた。

謝辞

本研究の一部は科研費基盤研究(C)15K00085の助成により行われた。

参考文献

[1] Kasahara, H., Honda, H., Mogi A., et al. :A multi-grain parallelizing compilation scheme for OSCAR (optimally scheduled advanced multiprocessor), *Fourth International Workshop Santa Clara* (1991).

[2] Kimura, K., Cecilia, G., Hayashi, et al. :OSCAR API v2.1: Extensions for an Advanced Accelerator Control Scheme to a Low-Power Multicore API, *7th Workshop on Compilers for Parallel Computing (CPC2013)* (2013).

[3] 佐藤卓也，見神広紀，林明宏ほか：OSCAR API標準解釈系を用いたParallelizable Cプログラムの評価，情報処理学会研究報告，Vol.2010-ARC-191-2 (2010).