発表概要

大規模データ処理向けElixirアプリケーションの高速化のためのOpenCLコードの生成手法

安部 竜矢[1,a)]　宮城 竜大[1]　高木 直史[1]　高瀬 英希[2,3]

2022年3月17日発表

Elixir は Erlang VM 上で動作する関数型言語であり，Ruby を参考とした言語設計がなされている．特徴として，動的型付け，データのイミュータブル性，および，map/reduce モデルによる並列処理の記述容易性などがあげられる．これらの特徴から，特に設計生産性向上の観点から Elixir を大規模な並列計算アプリケーションへ活用することの機運が高まっている．しかし，Elixir はプロセッサ上の仮想マシンで動作する方式であるため，性能面での本質的な限界がある．本研究の目的は，大規模なデータを扱う Elixir アプリケーションの高速化である．本論文では，map 関数を用いた Elixir コードから機能的に等価な OpenCL コードを生成する手法を提案する．OpenCL はデバイス移植性の高い API ライブラリであり，GPU や FPGA など種々のデバイスでアプリケーション高速化を図ることができる．コード生成の際に，型付け規則の違い，および，パイプ演算子に代表される Elixir の特徴的な記法が課題となる．本研究では，これらを解決するための中間表現および静的な型推論を設計する．生成されたコードは合わせて生成される Erlang の FFI である NIFs を用いた通信機構を介して GPU 上などで実行できるが，性能ボトルネックとなりうるデバイス間通信が多発することになる．この回数を削減するために，複数の map 関数をまとめる構文木の簡約方法についても提案する．提案手法に基づきコード生成系の実装を行い，大規模なデータを扱うベンチマークで評価を行い本提案の有効性を示す．

Presentation Abstract

A Method of Synthesizing OpenCL Code for Accelerating Large-scale Data Application by Elixir

Tatsuya Abe[1,a)]　Ryota Miyagi[1]　Naohumi Takagi[1]　Hideki Takase[2,3]

Presented: March 17, 2022

Elixir is a functional language that runs on the Erlang VM, and is designed Ruby-like. Elixir has features such as dynamic typing, immutability of data, and ease of describing parallel processing using the map/reduce programming model. From its features, the momentum for utilizing Elixir programming for large-scale parallel computing applications is increasing, especially from the viewpoint of improving design productivity. However, since Elixir runs on a virtual machine on the processor, there is an essential limit in terms of performance. The purpose of this paper is accelerating large-scale data application by Elixir. In this paper, we propose a method to generate functionally equivalent OpenCL code from Elixir code using the map function. OpenCL is an API library with high device portability, and can accelerate applications on various devices such as GPU and FPGA. When generating code, the differences in typing rules and the characteristic syntax of Elixir such as pipe operator become issues. In this study, we design intermediate representations and static type inference to solve these problems. The generated code can be executed on the GPU etc. via the communication mechanism using NIFs which is Erlang's FFI that is also generated. In order to reduce the number of device-to-device communication at that time, we also propose a method for reducing the syntax tree that merge multiple map functions. We implement a code generation system based on the proposed method and evaluate it with a benchmark using large-scale data to show the effectiveness of this proposal.

This is the abstract of an unrefereed presentation, and it should not preclude subsequent publication.

[1] 京都大学
Kyoto University, Kyoto 606–8501, Japan
[2] 東京大学
The University of Tokyo, Bunkyo, Tokyo 113–8654, Japan
[3] JST さきがけ
JST PRESTO, Kawaguchi, Saitama 332–0012, Japan
[a)] abe@lab3.kuis.kyoto-u.ac.jp

© 2022 Information Processing Society of Japan