## 発表概要

# 並列指向プログラミング言語の設計と実装

柴田 謙[1,a)]　高前田 伸也[2,b)]

2022年7月28日発表

近年，マルチコア CPU が主流となってきたが，現在主流の言語はシングルコアが主流の時代に設計された物が多く，並列実行を行って動作速度の向上を行うためには多くの場合で特別なライブラリや構文が必要になり，簡単に並列実行が行えるとはいえない．このマルチコアの利点を生かすため，特別な構文なしで並列実行を規定として行うプログラミング言語「Coa」を開発した．提案するプログラミング言語 Coa は，並列実行を既定とするが，単につねに並列実行を行うとするとデータレース問題が生じる．この問題を避けるため，CPU のアウト・オブ・オーダ実行がデータ間の依存関係を検出するのと同様に，変数の依存関係を自動検出し，データレースが生じないように自動的に実行順序と並列性を制御する．このため，内部で並列実行を行いつつも外から見た振舞いは逐次実行と同様となり，プログラムの複雑さを増やさずに実行速度を向上できる．また，処理単位が小さく逐次実行のオーバヘッドが大きい場合に備え，純粋でない関数と指定すれば逐次実行される機能もある．Coa のインタープリタは Go で書かれており，Coa で書かれたソースコードは，goroutine を用いてインタープリタ内で並列実行される．現在，基本演算，繰返し，条件分岐，関数定義などの基本的な機能があり，FizzBuzz 問題や情報オリンピックの問題等を解決できる程度の言語機能を実装している．比較実験として，13 ファイルのダウンロードとマンデルブロ集合を表示するための計算を，逐次処理と並列処理で実行した．それぞれの実行時間とコード行数を比較し，Coa はコードの複雑さを増すことなく並列実行して処理速度が上がることを確認した．本発表では，Coa の仕組みと特徴，今後の課題を説明する．

## Presentation Abstract

# Design and Implementation of a Parallel-native Programming Language

Ken Shibata[1,a)]　Shinya Takamaeda[2,b)]

Presented: July 28, 2022

Although multi-core CPUs have become mainstream in recent years, many of the current mainstream languages were designed when single-core CPUs were the mainstream, and often require special libraries and syntax to improve execution speed through parallel execution, making parallel execution not easy to achieve. We developed a parallel-oriented programming language, named Coa, which makes parallel execution the default. To avoid data races, it automatically detects variable dependencies and executes programs in parallel, just as out-of-order execution of CPUs detects dependencies among data. Therefore, although parallel execution is performed internally, the behavior seen from the outside is the same as sequential execution, and execution speed can be improved without increasing the complexity of the program. It also has a function that enables sequential execution, in case the processing unit is small and the overhead of sequential execution is large. The interpreter itself is written in Go, and source code written in Coa is executed in parallel in the interpreter using goroutine. Currently, language functions for solving FizzBuzz problems and information and opinion problems are being implemented. As a comparison experiment, downloading 13 files and computing to display the Mandelbrot set were executed in sequential and parallel processing. The execution time and number of lines of code for each were compared, and it was confirmed that Coa improves processing speed through parallel execution without increasing the complexity of the code. In this presentation, we will discuss the mechanisms and features of Coa.



[1] ウィリアム・ライオン・マッケンジー高等学校
William Lyon Mackenzie Collegiate Institute, Toronto, Ontario, Canada M4G2S1
[2] 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113–0033, Japan

[a)] kenxshibata@gmail.com
[b)] shinya@is.s.u-tokyo.ac.jp