ハイパフォーマンスプログラミング言語 Rust

TU 坂田誠也

お品書き

- 自己紹介
- Rust とは
- 開発パフォーマンス
- ランタイムパフォーマンス
- Rust のユースケース

お品書き

- **自己紹介**
- Rust とは
- 開発パフォーマンス
- ランタイムパフォーマンス
- Rust のユースケース

自己紹介

坂田誠也

技術的なこと

- だいたいなんでもやってる
- 最近は TypeScript+GCP

技術的じゃないこと

お品書き

- 自己紹介
- **Rust とは**
- 開発パフォーマンス
- ランタイムパフォーマンス
- Rust のユースケース

Rust とは

- Rust の紹介
- 機能・特徴

Rust とは

- **Rust の紹介**
- 機能・特徴
- Rust 製ソフトウェア

Rust とは

Rust の紹介

- Mozilla が支援しているオープンソースのシステムプログラミング言語
- 速度、並行性、安全性を保証する C 言語と C++に替わるプログラミング言語を目指している
- 強い型付けでマルチパラダイム

Rust とは

- Rust の紹介
- **機能・特徴**
- Rust 製ソフトウェア

Rust とは

機能・特徴

- 所有権・借用・lifetime
- null safe
- 総称性(generics)
- パターンマッチ
- GC なしでの自動での安全なメモリ管理
- マクロ
- 言語レベルで組み込まれているテスト機能
- 優秀なパケージマネージャー

Rust とは

- Rust の紹介
- 機能・特徴
- **Rust 製ソフトウェア**

Rust とは

Rust 製ソフトウェア

- Firefox
  - Servo(HTML レンダリングエンジン)
  - Quantum(ウェブエンジン)
- Redox(マイクロカーネル OS)
- Deno
- Read States(Discord のメッセージ基盤サービス)
- Nucleus(Dropbox の同期エンジン)

開発パフォーマンス

ランタイムパフォーマンス

Debian のプロジェクトが様々なプログラミング言語の実行速度を比較検証している。

The Computer Language Benchmarks Game

C++や C に次ぐ実行速度を誇る。

ランタイムパフォーマンス

Q. なんで速い？
A1. 機械語に直接コンパイルされるため
A2. ガベージコレクションを持たないため
A3. ゼロコスト抽象化によりランタイムシステムのオーバーヘッドを追求しているため

ランタイムパフォーマンス

機械語に直接コンパイル

Python や Java は独自の VM を介して VM 用の言語に変換される

➡️ 速度面で不利になる。

ランタイムパフォーマンス

機械語に直接コンパイル

C、C++や Rust はコンパイル後の最終結果は機械語になり、VM を介さずに実行できるので VM による速度低下が起きない。

➡️ 多くのプログラミング言語との速度の違い

GCC(GNU Compiler Collection)や LLVM(Low Level Virtual Machime)などによってプロセッサに応じた機械語を生成する。（Rust は LLVM を使用）

ランタイムパフォーマンス

ガベージコレクションがない

多くのプログラミング言語では、不要なメモリ領域を開放する処理を裏で逐一行う。

C や C++は手動でメモリを確保・開放する必要があるのでガベージコレクションは便利

ランタイムパフォーマンス

ガベージコレクションがない

ガベージコレクションの処理はいつ実行されるか分からず、不要なメモリを開放するのに処理の停止時間が発生する。

➡️ これが Go との速度の違い

ランタイムパフォーマンス

ゼロコスト抽象化

端的に言えば**抽象化した処理を実行するために必要な負荷を可能な限り小さくすること**（元は C++）

- 実行速度の低下
- メモリ使用量の増加

trait と dyn

クラスは内部に状態を持ち隠蔽することで外から見たときに内部の細かい処理に気を使わなくても良いようにする（カプセル化）