debug.gem によるRubyアプリケーションのデバッグ

笹田耕一

クックパッド株式会社
ko1@cookpad.com

今日のまとめ

- debug.gem 使ってみてください
  - **IDE 連携**便利です
    - VSCode, Chrome, …
  - **リモートデバッグ**便利です
    - 別プロセス
    - 別ノード（コンテナ）
- ソコソコ難しいので、お気軽にお問い合わせください

debug.gem

- Ruby 3.1 から同梱された最新のRuby用デバッガ
  - gem install debug で Ruby 2.7 以降で利用可能
- 特徴
  - 速い（既存のものに比べて、一部機能）
  - **IDE連携**
  - **リモートデバッグ**
- 参考資料
  - https://github.com/ruby/debug/ （in English）
  - https://techlife.cookpad.com/entry/2021/12/27/202133 （日本語）

プログラムのデバッグ

例えば…

- 開発時：**なぜか**おかしな挙動をする
- 開発時：**なぜか**反応がない
- 運用時：**なぜか**遅い

debug.gem の基本機能
だいたいほかのデバッガと同じ

- プログラムを途中で**一時停止**する
  - 停止位置をプログラムに埋め込んで止める（debugger メソッド）
  - ブレイクポイントを設定し、止める（行、メソッド、例外など）
  - Ctrl+C（シグナル受信時）で止める
  - リモートデバッガで接続して止める
- プログラムを**ちょっとずつ動かす**
  - ステップイン、ステップオーバー、ステップアウト
  - 再開
- プログラムの**状態を調べる**
  - バックトレース、ローカル変数、…
  - 任意の式の実行
- その他（record & replay, logging, etc.）

debug.gem の利用方法（UI）

- コマンドラインインターフェース（CUI）
  - 独自の操作コマンド体系
  - gdb, lldb など既存のデバッガに近い
- IDE 連携（VSCode, Chrome）
  - （だいたい）マウスクリックで操作可能
  - わかりやすいのでオススメ

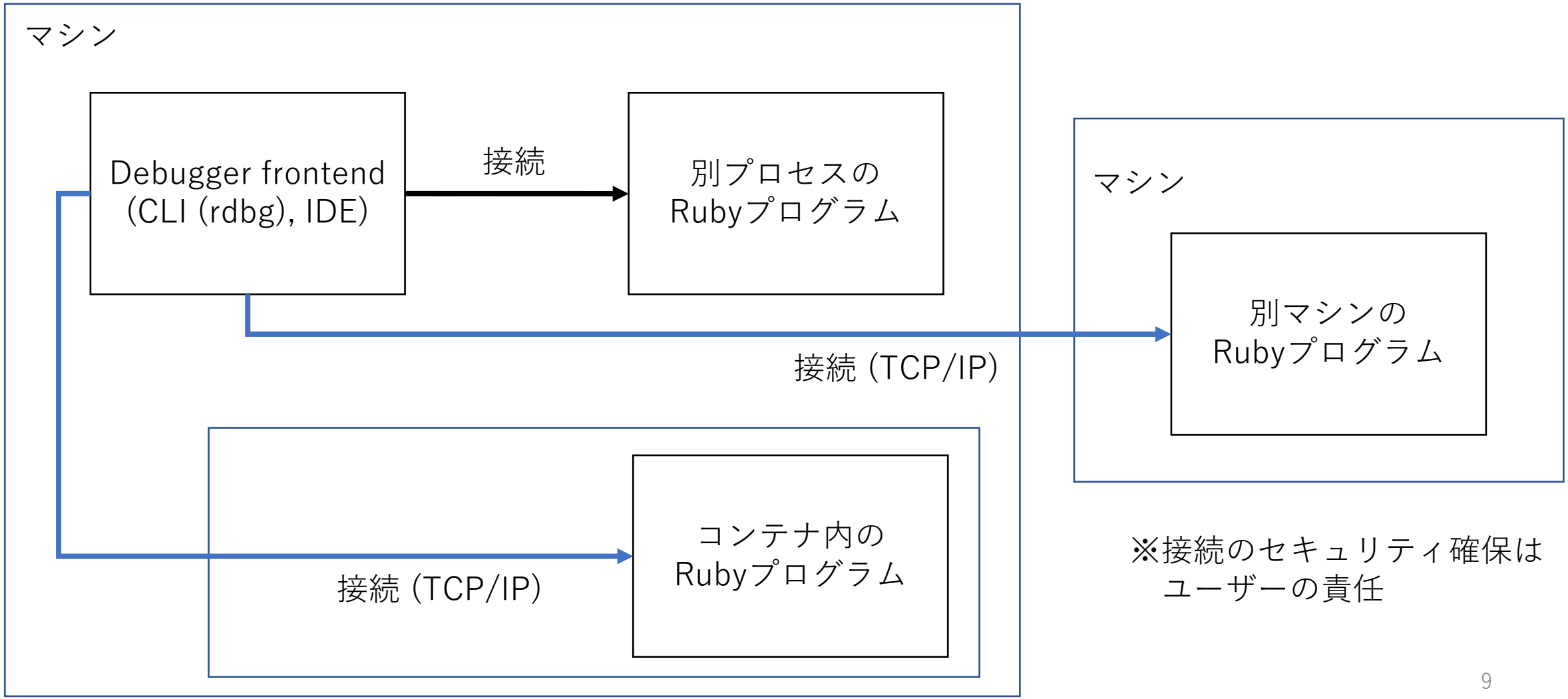
debug.gem の利用方法（起動方法）

- プログラムを指定してデバッガ（CLI）とともに起動
  - `rdbg script.rb`
  - `rdbg -c -- bundle exec ruby script.rb`
- スクリプトに記述して起動
  - `require 'debug/start'`
- IDE から起動
  - VSCode (rdbg Ruby Debugger 拡張が必要)

リモートデバッグ

- すでに起動しているRubyアプリにあとから接続しデバッグ
  - デバッグポートをあけて起動しておき、そこにデバッガが接続
- 端末（tty）で接続できないとき便利
  - 別コンテナ・別マシン（ノード）で起動しているとき
  - デーモンプロセス（railsサーバも）
  - リダイレクトしているとき（`ruby app.rb | foo`）
- デバッグしたいわけじゃないけど、プロセスの状況を監視したいときにも便利

リモートデバッグ



※接続のセキュリティ確保は
ユーザーの責任

デバッガ活用のストーリー
(1) 開発時（その１）

1. プログラムを開発
2. テストが失敗するが、ぱっと見ても修正点がわからない
3. 失敗する直前に**ブレイクポイント**を設定し、**デバッガ上でテスト再実行**
4. ブレイクポイントで止め、**ステップ実行**しながらプログラムの**状態を観察**しながら進めていく
5. おかしな状態を発見して修正

- 「printデバッグ」よりも、随時確認できるので良い
- 「`binding.irb/pry`」よりも、ステップ実行等できるので良い

デバッガ活用のストーリー
(1) 開発時（その２）

1. プログラムを開発
2. 試しに**リモートデバッグポートを開けて**実行してみたら、プログラムから反応がない
3. **リモートデバッガを接続**
4. 接続したタイミングでプログラムが一時停止されるので、プログラムの状態を確認し、問題を把握

- （デバッグポートが開いていれば）簡単にデバッガで接続可能

デバッガ活用のストーリー
(2) 運用時

1. 運用しているプログラムが重いが状態がよくわからない
2. **リモートデバッグポート**をあけて本番環境で実行
3. プログラムが重い状態を確認後、**デバッガで接続**し、プログラムの状態を確認

- プロファイラと比較して、より詳細な情報の取得が可能

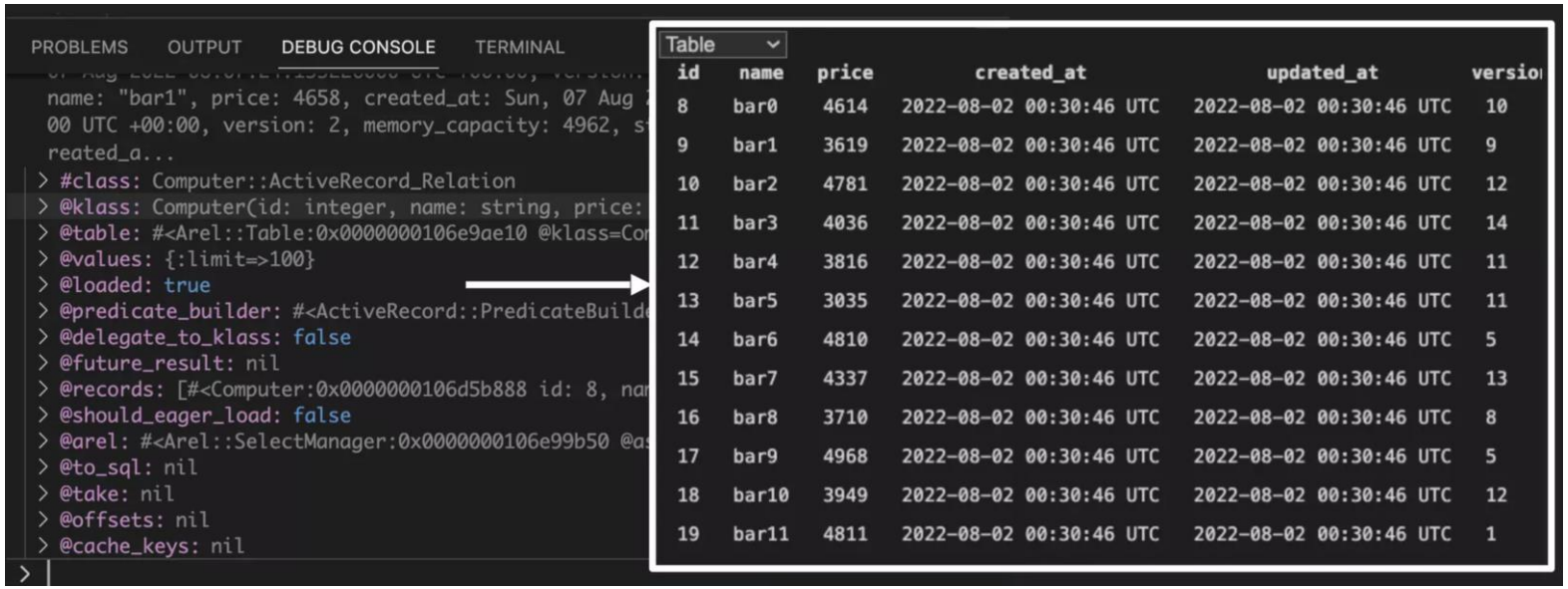
デバッガ活用のストーリー
(3) コードリーディング時

1. よくわからないプログラムの解読作業
2. **とりあえずデバッガ**で1行ずつ進めていく
   - 興味あるところはステップインで
   - 興味ないところはステップアウト / ステップオーバーで
   - わかっていれば、興味あるところにブレイクポイントを設定し、そこまで実行
   - ロギング機能を使って興味あるところの実行の流れを確認
3. 途中、ローカル変数などの**状態を確認**しながら進める

- 「読む」だけよりも、正確にプログラムの処理を確認可能

今後の発展

- よりリッチなIDE体験
  - Tools for Providing rich user experience in debugger by Naoto Ono, RubyKaigi 2022



でも、デバッガって難しいんでしょう？

- **はい**、すべて完全に使いこなすのは難しいです
  - ドキュメントに全部書いてあります
  - が、機能が豊富なので、完全な活用は難しいです。特に**起動方法**
  - いつかはわかりやすい how to を作りたいと思っています
- **お気軽にお問い合わせください**
  - debug.gem チョットデキル人材です。困ったらお声かけください
  - デバッガに必要な機能のサーベイになるので私にもメリット
  - 実際のコードをベースに議論することも可能

今日のまとめ

- debug.gem 使ってみてください
  - **IDE 連携**便利です
    - VSCode, Chrome, …
  - **リモートデバッグ**便利です
    - 別プロセス
    - 別ノード（コンテナ）
- ソコソコ難しいので、お気軽にお問い合わせください