【別添２】

Ruby プログラムを高速に実行するための処理系の開発
― YARV: Yet Another RubyVM ―

１．背景

日本発のオブジェクト指向スクリプト言語 Ruby は、その使い易さから多くのユーザを擁し、世界的に広く利用されている。しかし、現在の Ruby 処理系の実装は、単純な構文木をたどるインタプリタであるため、その実行速度は遅い。これを解決するために幾つかのバイトコード実行型仮想マシンが提案・開発されているが、たとえば Ruby のサブセットしか実行できない、実行速度が十分でない、などの問題がある。

２．目的

このような背景から、本プロジェクトでは、Ruby プログラムを正しく高速に実行可能な処理系を開発し、「Ruby は遅い」という悪評を打破することを目的とする。具体的には、さまざまな最適化を実装し、高速化を実現する。また、実験的に JIT（Just-in-Time）コンパイラや AOT（Ahead-of-Time）コンパイラにも挑戦し、さらなる性能向上の可能性を探る。

３．開発の内容

開発は、まず Ruby プログラムを十分に表現可能で、効率的に実行できる命令セットの設計を行い、これに基づいて実行に必要な各種処理系を実装した。

(1) Ruby プログラムを表現可能な命令セットの設計
(2) Ruby プログラムを(1)の命令セットへ変換するコンパイラの開発
(3) (1)の命令セットを効率的に実行する仮想マシンの開発
(4) JIT コンパイラの開発
(5) AOT コンパイラの開発
(6) 性能評価

従来の Ruby 処理系は、構文木を直接実行するインタプリタであるのに対し、YARV ではスタックマシンを採用した。その違いを図１に示す。

YARV の全体構成、および動作の流れを図２に示す。

開発した処理系においての性能評価の結果、Ruby の各機能は 10 倍もの性能向上を実現できた。また、ベンチマークプログラムでは 2 倍以上の性能向上を確認できた（詳細は図５を参照）。

以下、(1) ～ (6) について示す。

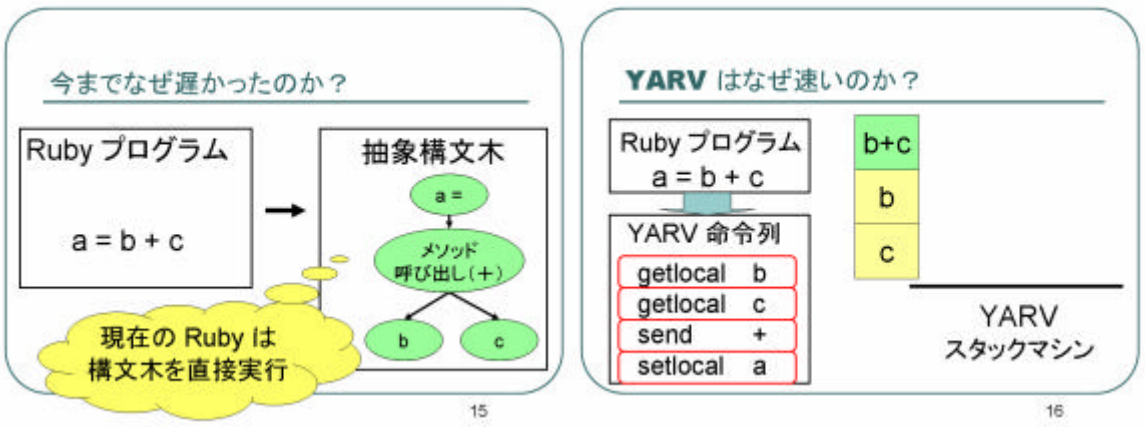


図１　現在の処理系と YARV の違い

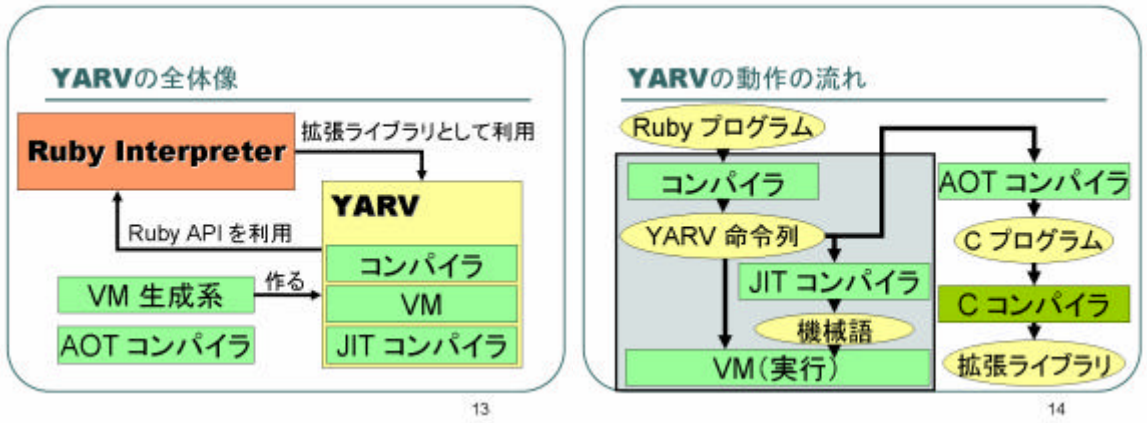


図２　YARV の全体像、および動作の流れ

(1) Ruby プログラムを表現可能な命令セットの設計

Ruby プログラムを十分に表現でき、かつこれを実行する処理系が効率良く処理することのできる命令セットを設計した。

なお、命令記述フォーマットを定義し、各命令をこのフォーマットで記述した。この命令記述フォーマットを採用することで、(3)の仮想マシン（VM）などの実装が非常に楽になった。

(2) Ruby プログラムを(1)の命令セットへ変換するコンパイラの開発

Ruby プログラムを構文解析して作られる構文木を対象に YARV 命令列に変換するコンパイラを開発した（図３）。

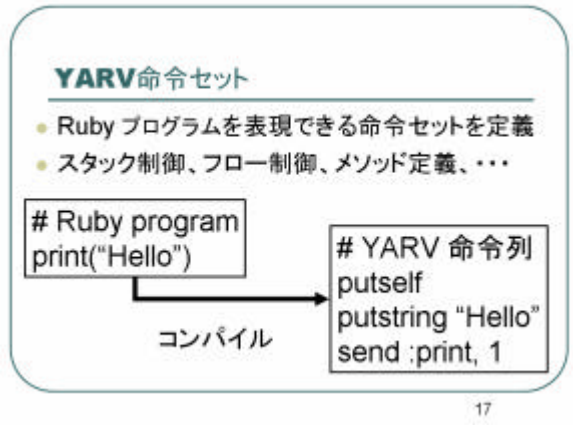


図３　コンパイラの働き

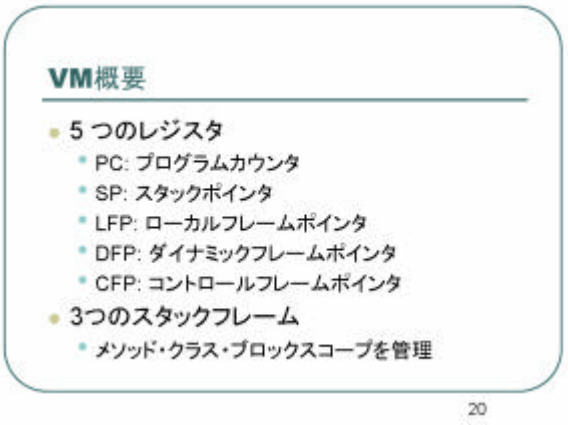


図４　VM の概要

(3) (1)の命令セットを効率的に実行する仮想マシンの開発

YARV 命令セットで表現された YARV 命令列を実行するスタックマシンモデルの仮想マシン（VM: Virtual Machine）を開発した（図４）。この VM には高速化のためのいろいろな最適化手法を適用している（ダイレクトスレッデッドコード、2-level スタックキャッシング、特化命令、オペランド融合、命令融合、インラインキャッシュの利用など）。

(4) JIT コンパイラの開発

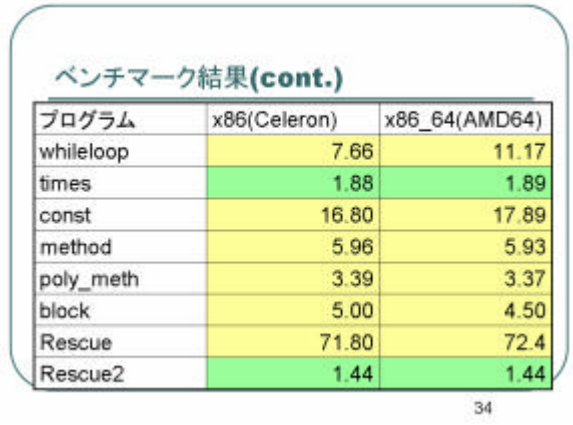
YARV 命令列をネイティブコード列に変換する実験的な JIT コンパイラを開発した。

(5) AOT コンパイラの開発

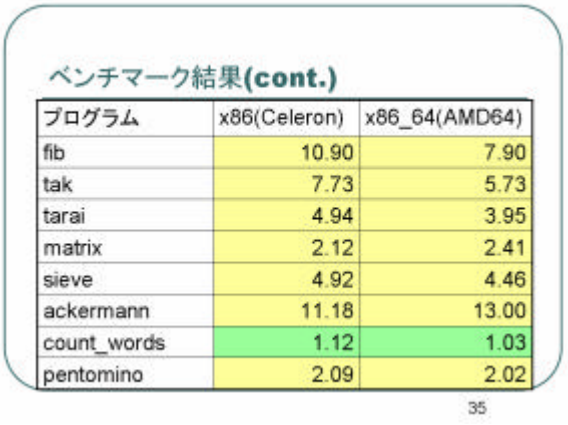
YARV 命令列を C 言語に変換する実験的な AOT コンパイラを開発した。

(6) 性能評価

図５に示すような性能向上をベンチマークプログラムによって確認した。



プログラム	x86(Celeron)	x86_64(AMD64)
whileloop	7.66	11.17
times	1.88	1.89
const	16.80	17.89
method	5.96	5.93
poly_meth	3.39	3.37
block	5.00	4.50
Rescue	71.80	72.4
Rescue2	1.44	1.44



プログラム	x86(Celeron)	x86_64(AMD64)
fib	10.90	7.90
tak	7.73	5.73
tarai	4.94	3.95
matrix	2.12	2.41
sieve	4.92	4.46
ackermann	11.18	13.00
count_words	1.12	1.03
pentomino	2.09	2.02

図５　性能評価結果（ベンチマーク結果）　結果は既存の処理系との速度向上比

また、開発にあたっては 100 以上のテストケースを用いて Ruby 処理系としての動作を確認し、品質管理を行った。

４．従来の技術（または機能）との相違

本プロジェクトの最大の成果は、高速化である。YARV では、一から Ruby 処理系（実行部分）を作り直したが、様々な最適化を盛り込むことで実行効率を大幅に向上することができた。具体的には、既存の Ruby と比較して基本性能で 10 倍を実現した。また、既存の Ruby の分かり難いソースプログラム（Ruby の作者のまつもとゆきひろ氏曰く「魔法みたいな処理系のソースコード」）を一新し、新しく作り直したことで処理系の見通しが付き易くなった。これらの成果は、まつもと氏からも評価されており、次期 Ruby の実装として取り込まれる見通しである。

他の言語の処理系と比較すると、例えば同じくオブジェクト指向言語である Python の実行処理系は、現在では最適化にあまり力を入れていない。そのため、言語設計のレベルでは Ruby よりも Python の方が有利であるものの、処理系の性能では Ruby の方が勝っている。Java の VM や.NET 環境、Perl 6 のために開発が進められている Parrot と比較した場合、YARV では Ruby に特化している分、Ruby に適した最適化を行うことができた。

５．期待される効果

Ruby プログラムが高速に実行されることによって Ruby 自体の価値が向上し、利用者増加が見込まれる。これは、日本発のソフトウェアが世界中で利用され、評価されることと同意である。また、すでに Ruby を利用している人にとってはさらに利用用途が広がり、たとえば今までは性能的な問題のあった大規模ウェブアプリケーションにも利用可能になる。

６．普及（または活用）の見通し

まつもと氏より、Ruby の公式バージョンとして開発に取り組まないか、との打診を受けており、現在交渉中である。このまま順調に行けば、本プロジェクトの成果は次期 Ruby の処理系（Ruby 2.0 Rite）に正式に採用される予定である。Ruby 2.0 としてリリースされると世界中で何万人規模（もしくはそれ以上）の Ruby ユーザに活用されることとなる。

７．開発者名（所属）

笹田耕一（東京農工大学大学院 / 日本 Ruby の会）

（参考）

YARV: Yet Another RubyVM 公式ページ: http://www.atdot.net/yarv/