YARV
Yet Another RubyVM

http://www.atdot.net/yarv/

ささだ　こういち
東京農工大学大学院
ko1 at atdot dot net

Lightweight Language Weekend 2004 – Lightning Talk 資料

背景

- オブジェクト指向スクリプト言語Ruby
  - 使いやすいと評判
  - 世界中で広く利用
  - 日本発（主開発者：まつもとゆきひろ氏）
- 既存のRuby処理系は遅い
  - 構文木を辿って実行(eval関数の再帰呼び出し）
  - パフォーマンスの問題から、泣く泣く Javaなどを選択
  - 都市伝説化

    「Rubyって遅いんでしょ？　使えないよ」

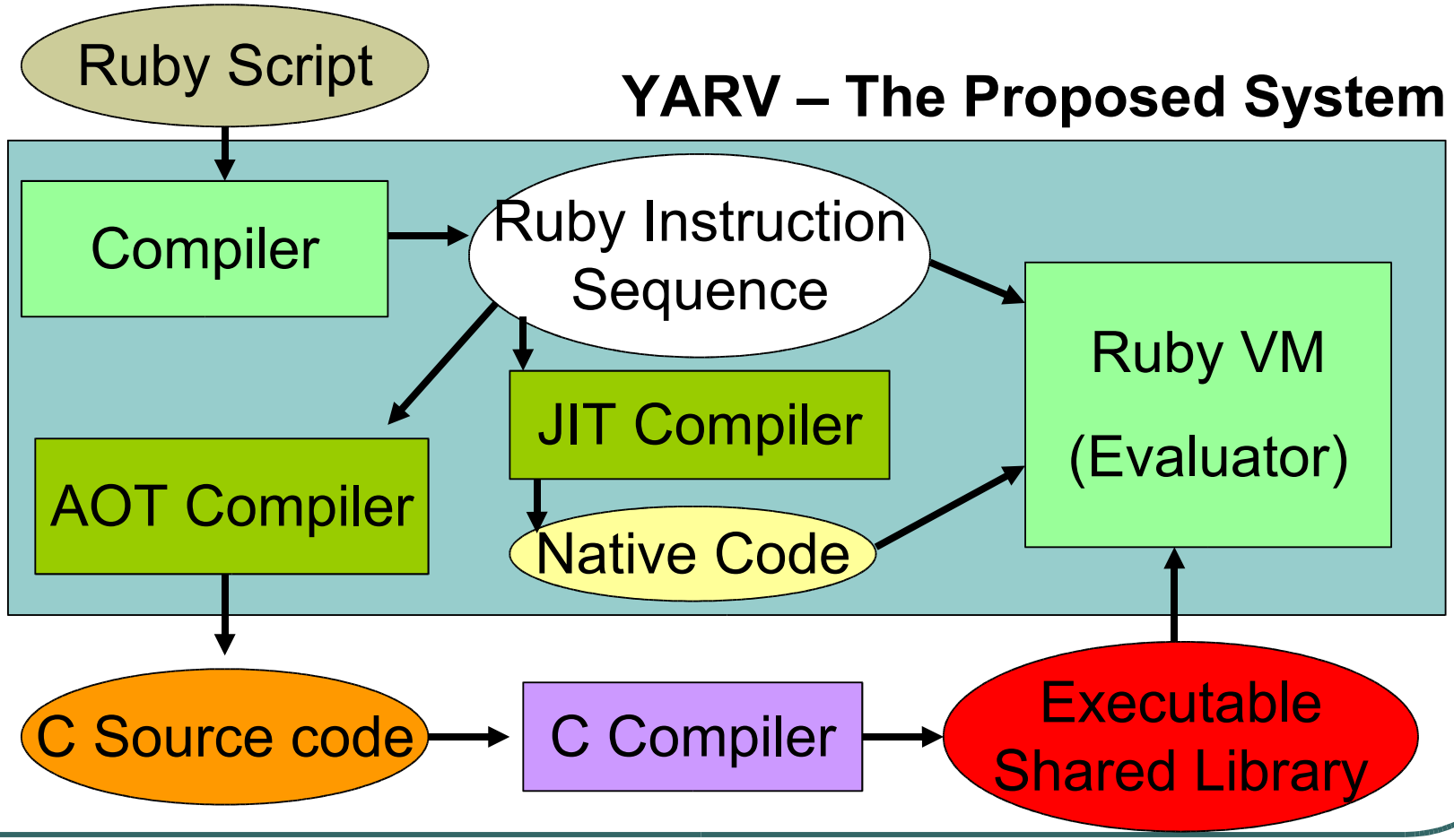
背景(cont.)

- Ruby処理系の処理速度向上が必須
  - バイトコード（仮想マシン型）処理系が最適
  - Lisp, Java, .NET, Smalltalk 処理系など
- いくつかのRubyバイトコード処理系→不十分
  - まつもと氏 平成12年度未踏ソフトウェア創造事業
    - 不完全（命令が不十分・コンパイラも無い）
  - ByteCodeRuby(George Marrows氏)
    - 現状のRuby処理系よりも遅い
  - その他、ほとんど作りかけ

YARV - Yet Another Ruby VM

- VM命令セットの設計
- コンパイラの設計・開発
- 仮想機械(RubyVM)の設計・開発
- JIT(Just In Time)コンパイラの設計・開発
- AOT(Ahead of Time)コンパイラの設計・開発
  - JIT/AOT Compiler は実験的
- 成果はオープンソースソフトウェアとして公開

提案するシステムの全体像

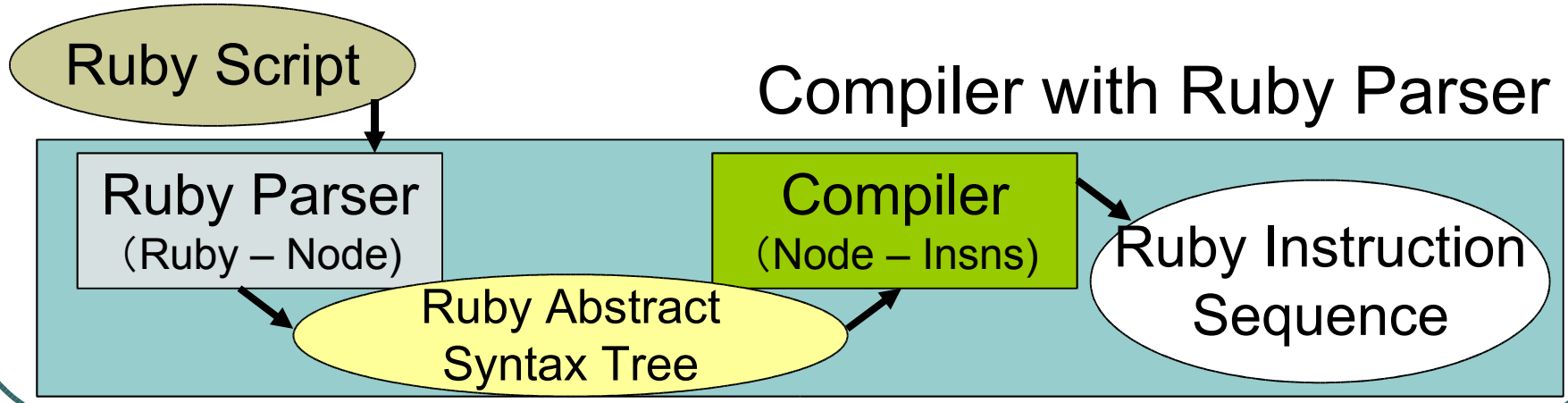


目標

- 世界一速いRuby処理系
  - 基本性能で2倍、JITコンパイラでは5倍
    AOTコンパイラでは10倍の性能向上を目指す
- 世界中の人に使ってもらえるように
  - **RubyVM** としての十分な機能を実装
- 次期Rubyの公式実装 Rite に
  - Ruby2.0 処理系 **Rite** （現在 1.9.0 開発中）
  - 本提案が成功すればこれを**Rite**として採用

開発方針

- 既存のものをなるべく利用
  - オブジェクトの管理（GC, etc）
  - Ruby Script パーサ
- Rubyの拡張ライブラリとして実装



技術的なこと

- バイトコードではなくワードコード
- 命令記述言語の作成
  - Cソースを元に、いろいろと変換ができるように
- 提案されている各種高速化技法の利用
  - 命令融合、オペランド融合
  - 2 level スタックキャッシング
  - ダイナミックレプリケーション
  - これらの自動的適用の検討（オリジナル）
- 既存のライブラリの利用の検討
  - libjit, lightningなど

プロジェクトの現状

- VM命令セットの基本部分を設計
- 基本部分を動作させるためのVMを開発
  - 簡単なプログラム → 2～3倍ほど性能向上
  - 動かないプログラム・速くならないプログラム多数
    - 定数の管理・ブロックの実行など
- メーリングリストにて議論
- ウェブページで公開中
  - YARV: Yet Another Ruby VM（英語）
    http://www.atdot.net/yarv/

以下，参考資料

Ruby Program

```
class Animal
  def bark
    raise “unimplemented”
  end
End
class Dog < Animal
  def bark
    puts “bow wow”
  end
end
```

VM実現方式比較

- スクラッチで書く
  - Pros.
    - Rubyに特化した命令・VMデザインが可能
  - Cons.
    - いろいろ大変（JITコンパイラなどの用意が特に）
- 既存VMに載せる(JavaVM / .Net)
  - Pros.
    - いろいろ楽
  - Cons.
    - Rubyに特化した命令ではなく、十分な最適化ができない
      - 特にメソッドディスパッチ
    - あまりおもしろくない
      - Ruby – 既存のVMモデルのマッピングに終始

VM実現方式比較(cont.)

- Stack Machine VM(本提案アプローチ)
  - Pros.
    - コンパイルが楽（スクリプト言語に最適）
  - Cons.
    - 最適化が制限される？　難しい？
- Register Machine VM(Parrot)
  - Pros.
    - 既存のコンパイラ技術による最適化が可能
    - 命令の移動が可能
  - Cons.
    - コンパイルに時間がかかる（レジスタ割付など）
- VM Generator – もっと最適化をがんばりたい

命令記述言語

例:

```
DEFINE_INSN # pragma
putstring   # insn name
(VALUE val) # operand
()          # pop from stack
(VALUE val) # push to stack
{           # insn logic
  val = rb_str_new3(val);
}
```

おわり