

The Object-Oriented Scripting Language

Ruby

<http://www.ruby-lang.org/>

Ruby is the interpreted scripting language for **quick** and **easy** object-oriented programming. It has many features to process text files and to do system management tasks (as in Perl). It is simple, straight-forward, extensible, and portable.

Oh, I need to mention, it's totally free, which means not only free of charge, but also freedom to use, copy, modify, and distribute it.

The creator of Ruby is **Yukihiro Matsumoto, a.k.a Matz**.

Quoted from "What's Ruby" page on <http://www.ruby-lang.org/>.

Ruby

Language Introduction

Dynamic OO Language Features

```
# Sample Ruby Script
class SampleClass
  def methodA
  end
end

class SubClass < SampleClass
  def methodB
  end
  def methodA
    # overload
  end
end

module SampleModule
  def methodX
  end
end

obj = SubClass.new

class SampleClass
  def methodC
  end
  include SampleModule

  if cond
    def methodD
    end
  end
end

1.object_id
eval("...")
```

Class Definition

Instance Method Definition

Subclass Definition

Module Definition

Instantiate SubClass Object

Class Re-open

Include SampleModule. Instance of SampleClass can use SampleModule's methods

All Definition statements are Execution statements (Can't do static analysis)

All values are objects (no primitive types)

Useful Evil Eval (evil for VM designer) (Evaluate any Ruby program on any context)

Other Ruby Features

- Mark and sweep garbage collection support
- Closure support
- Invoking a method with a block

```
# invoke method with a block
recv.method(args) do |params|
  body
end
```

```
; in Scheme
(method recv args
 (lambda (params)
  body))
```

- Exception Handling

```
# Ruby exception handling
begin
  # ...
  #
  rescue SomeError => e
  # ...
  #
  rescue => e
  # ...
  #
  ensure
  # ...
end
```

```
// Java exception handling
try{
  // ...
}
catch(SomeError e){
  // ...
}
catch(Exception e){
  // ...
}
finally{
  // ...
}
```

- Many useful libraries
- Networking, Distributed Computing, CGI
- Text Processing, XML, YAML
- System Management
- Design Pattern, Algorithm, Mathematics
- User-level (OS independent) Thread support
- Foreign Function Interface (Ruby C API)
- Highly portable



Ruby on Rails

Rails is a full-stack, open-source web framework in Ruby for writing real-world applications with joy and less code than most frameworks spend doing XML sit-ups (quoted from <http://www.rubyonrails.org/>).

Fifth International Ruby Conference a.k.a. RubyConf 2005 14-16, Oct. @ San Diego

Program of RubyConf2005

Friday, October 14

Panel 1

- Lucas Carlson: Advanced Classification in Ruby and Rails
- Akira Tanaka: open-uri, easy to use and extensible virtual file system

Panel 2

- Charles Nutter: JRuby: A Ruby VM in Java
- Koichi SASADA: YARV Progress Report
- Eric Hodel: Reimplementing Ruby

Yukihiro "matz" Matsumoto Roundtable

Saturday, October 15

Panel 3

- Kevin Baird: Refactoring No Clergy
- Brent Roman: Embedding Ruby into a Robotic Marine Laboratory
- Austin Ziegler: PDF::Writer

Panel 4

- Ryan Davis: Polishing Ruby: Power Tools and Toys
- Jim Weirich: Creating Domain Specific Languages in Ruby
- Karlin Fox System: Testing in Ruby with Systir

Keynote address: Yukihiro "matz" Matsumoto

Sunday, October 16

Panel 5

- David Heinemeier Hansson: The State of the Rails
- Nathaniel Talbott: Rails: Serving the long tail in 1883 and 2005
- Aslak Hellesoy: Continuous Integration with DamageControl

Workshops (details TBA)

- David Heinemeier: Hansson Hands-on Rails
- Jim Weirich & Chad Fowler: Continuations Demystified

YARV: Yet Another RubyVM

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

Innovating the Ruby Interpreter

Project Objective

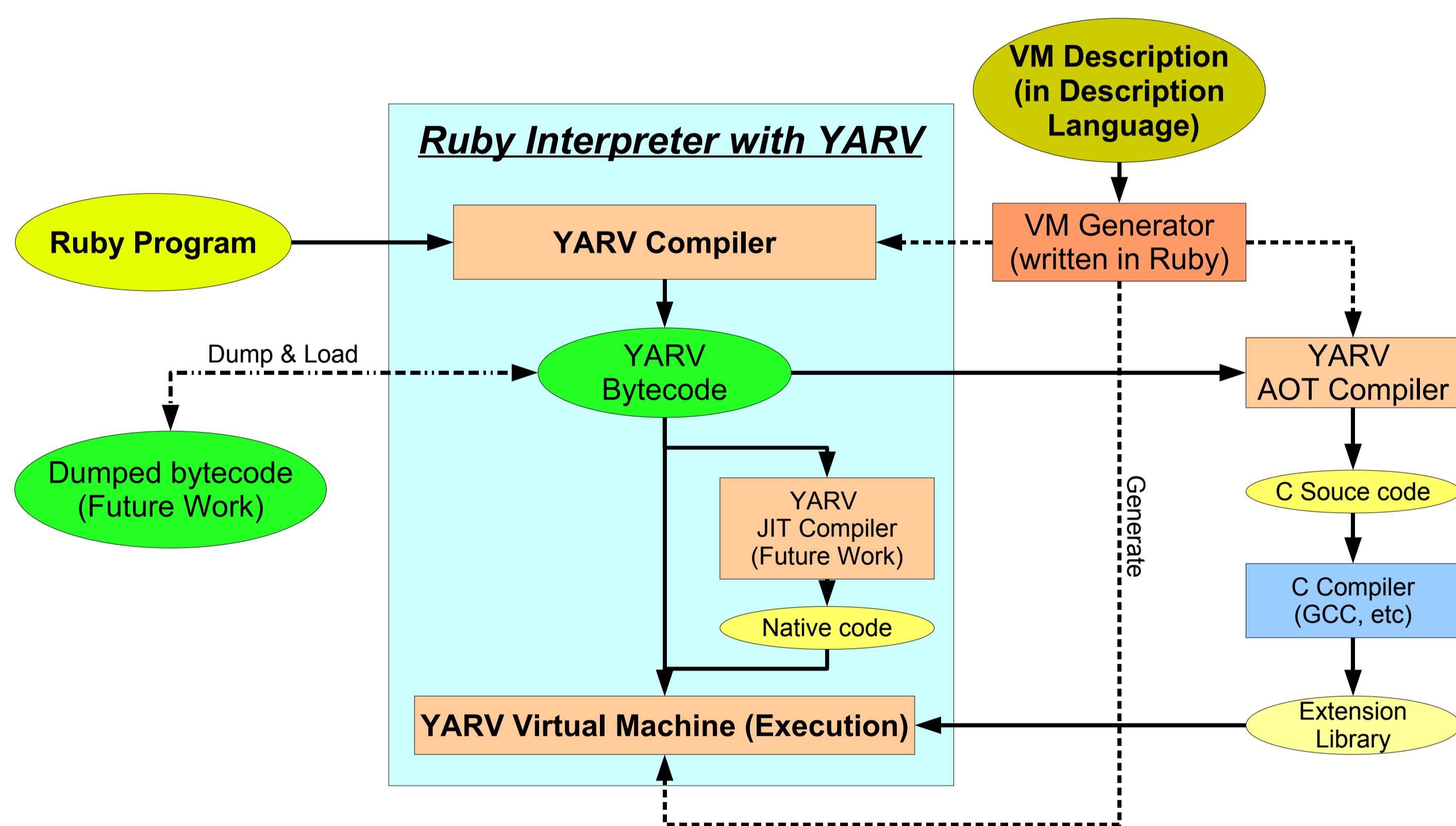
Ruby - an Object-Oriented scripting language - is used world-wide because of its ease of use.

However, **the current interpreter is slow** because it works with traversing abstract syntax tree. To solve this problem, several virtual machines have been developed, but none has achieved adequate performance or functionality. **YARV (Yet Another Ruby VM)** is new implementation of **Stack VM-based Ruby interpreter** that specifically aims at **high-speed execution**, overcoming weakness of existing implementations.

By **Koichi Sasada**

Nihon Ruby no Kai (Ruby Association in Japan) /
Graduate School of Technology,
Tokyo University of Agriculture and Technology
2-24-16 Nakach, Koganei-shi, Tokyo, Japan
E-mail: sasada@namikilab.tuat.ac.jp

Overview of System



Optimization for VM Speedup

- Direct Threaded Code
 - Well known VM speedup technique on GCC

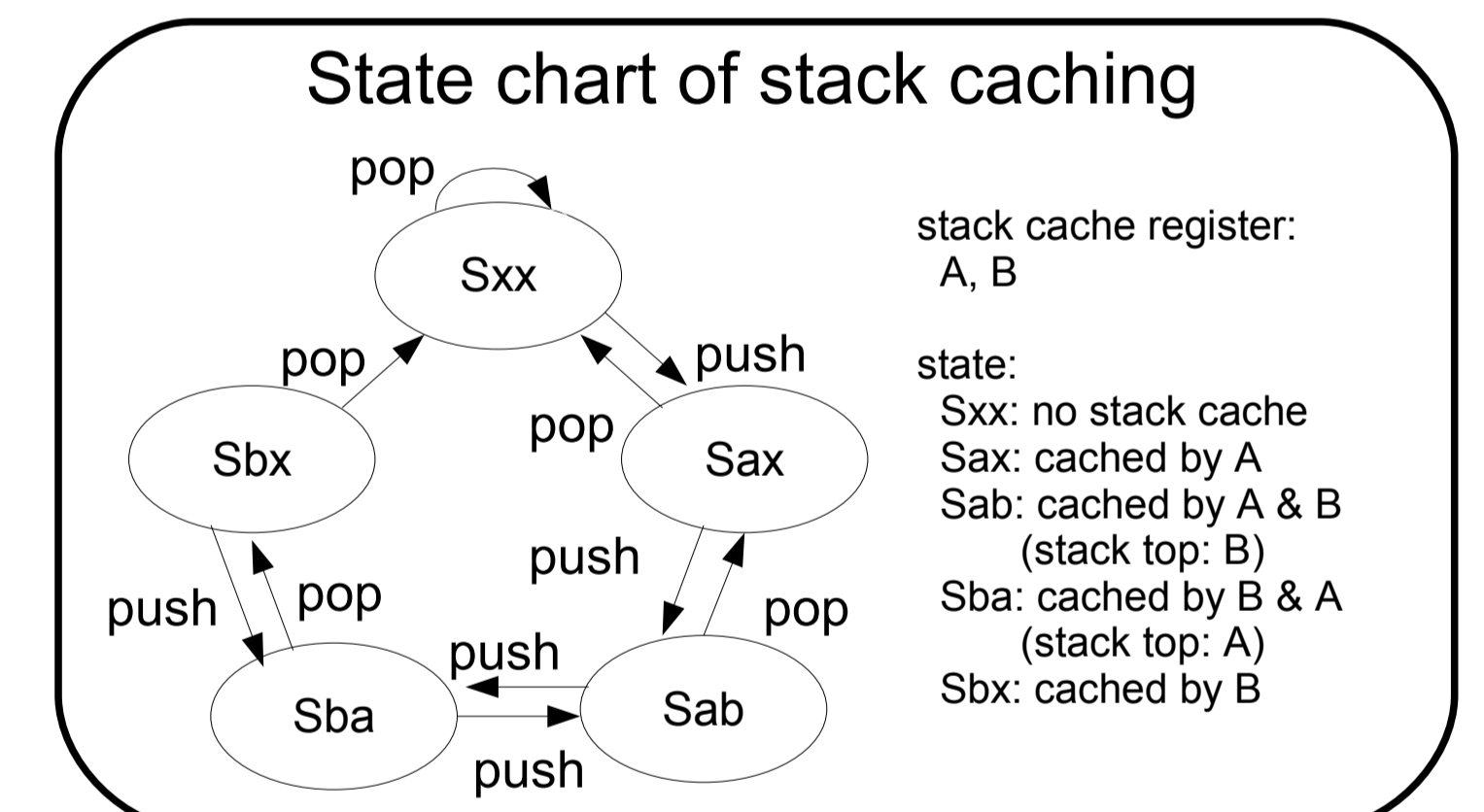
- Specialized Instruction
 - Ruby does not have primitive types
 - If a method selector is specific one, use special instruction instead of method dispatch

```
// Specialized "+" instruction
instruction opt_plus(x, y){
  if(x is Fixnum && y is Fixnum){
    if(Fixnum#+ is not re-defined){
      return x+y;
    }
  }
  return x.+(y);
}
```

- Operands, Instructions Unification (a.k.a. Super Instruction)
 - Create new instructions which embed specific operands or combine some instructions.
 - These instructions are auto generated by VM generator

- In-line Cache
 - In-line Method Cache
 - In-line Value Cache
 - For constant value access

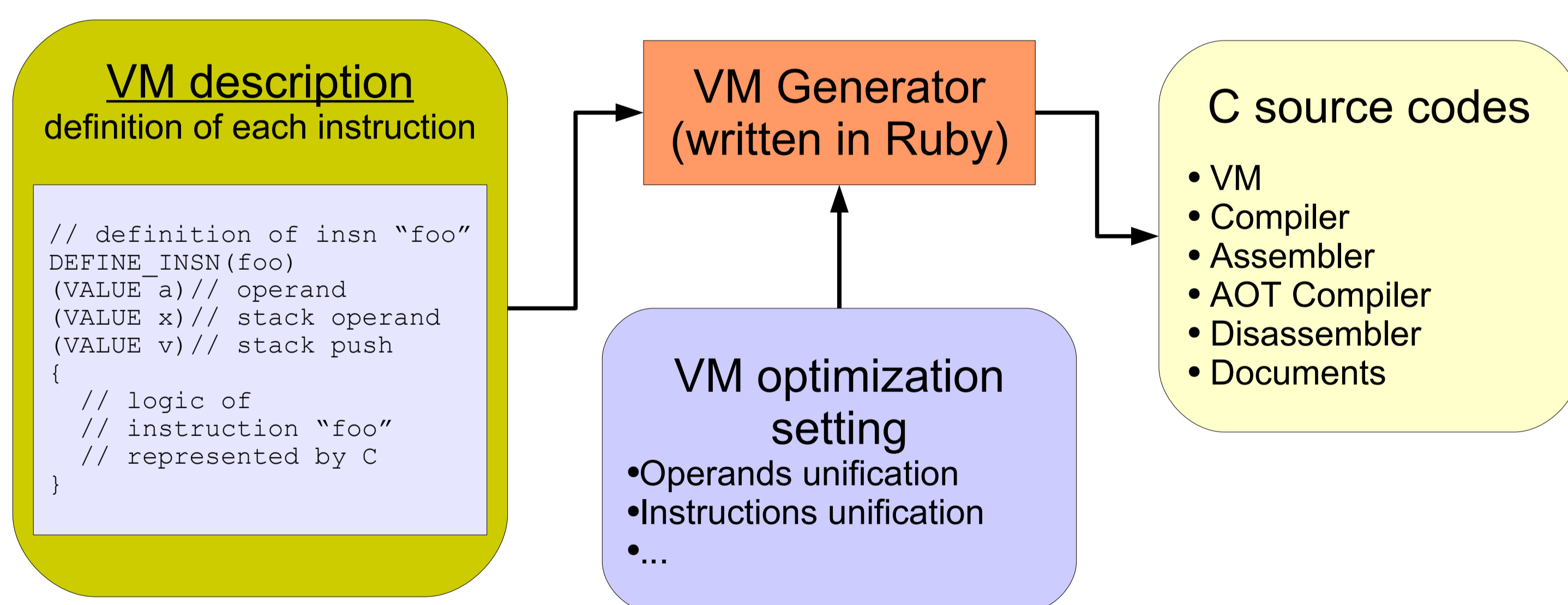
- Static Stack Caching
 - 2 registers, 5 states
 - Instructions for stack caching are auto generated



- Profiler
 - Instruction frequency
 - Operands frequency
 - For Operands unification
 - Instruction coupling ratio (using bi-gram)
 - For Instruction unification

- Native Compiler
 - JIT Compiler (future work)
 - AOT Compiler (YARV instructions to C)

VM code generation



This work flow is similar to vmgen (by M. Anton Ertl).

Evaluation

The two graphs on the right show results of YARV speedup ratio compared to current Ruby interpreter.

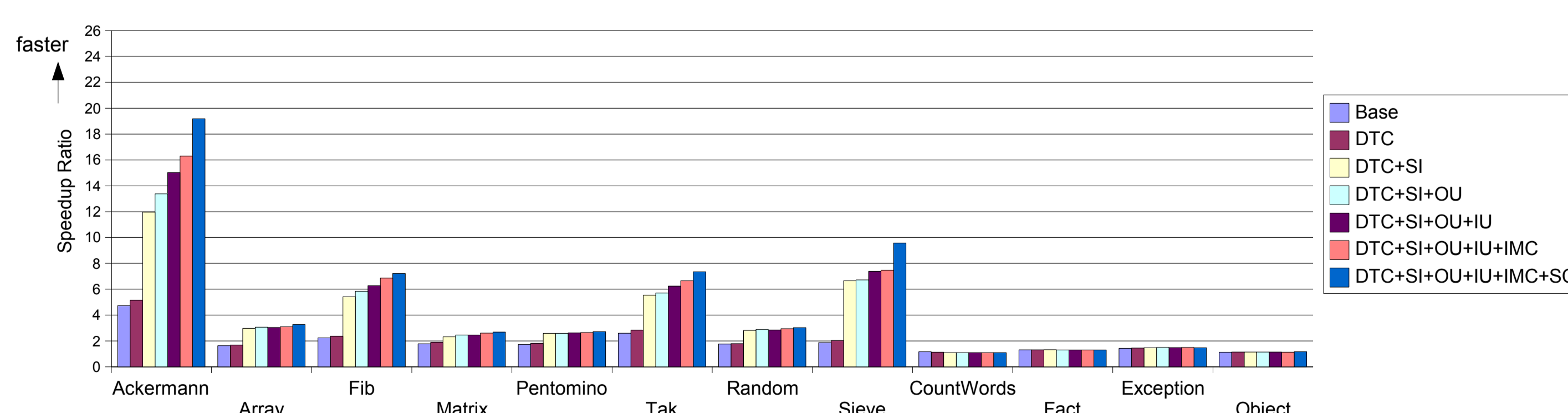
These bars show the results of the following optimizations applied incrementally:

- Base: only Base VM
- DTC: Direct Threaded Code
- SI: Specialized Instruction
- OU: Operands Unification
- IU: Instructions Unification
- IMC: In-line Method Cache
- SC: Stack Caching

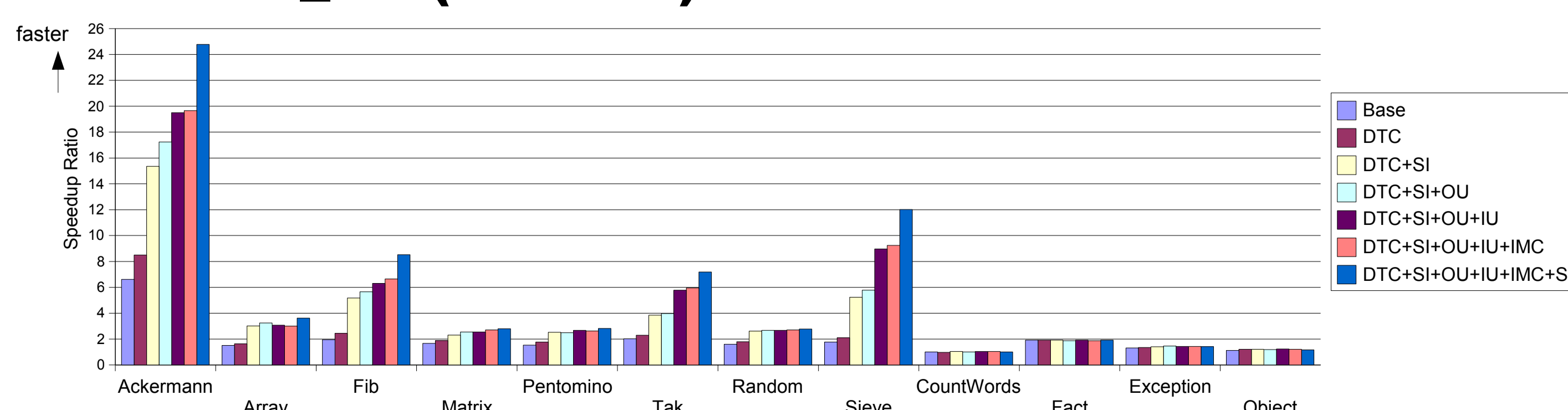
The results show that the speed of VM-dependent applications is greatly increased. However, other applications (those depending on GC, Object allocation, Regexp, and Bignum) are not significantly affected by YARV.

Optimizations are effective. Especially Ackerman's application can run 25 times faster than the current Ruby interpreter.

on x86



on x86_64 (AMD64)



Status & Future work

The grand goal of this project is to be **Ruby 2.0 Rite**.

Current YARV (version is 0.3.2) supports almost all Ruby syntax, native thread (only run concurrent), and Ruby C API. Of course, YARV is Open Source Software.

Issues:

- Support all ruby features (security model, etc)
- Multi-VM instance (like Java MVM)
- Native Thread support and run Ruby programs in parallel
- JIT (Just-in-Time) Compilation
- Complete AOT (Ahead-of-Time) Compilation
- Apply other optimizations using examples from Smalltalk, Self and other dynamic OO languages

Acknowledgments

This project is assisted by Exploratory Software Project 2004 (youth) and 2005 from IPA (Information-technology Promotion Agency, Japan). I thank Gyoung-Yoon Noh, Daniel Amelang, Wilson Bilkovich, Michiaki Baba, Shiro Kawai, Shashank Date, yarv-dev, yarv-level ML subscribers and all rubyists.