The Future of Ruby(3)

Heroku
Ruby Association

Yukihiro "Matz" Matsumoto
まつもとゆきひろ
@yukihiro_matz

您好

RubyConf China

Ruby

Ruby is Good

- Productive
- Flexible
- Fun

Ruby is Bad

"Ruby is Dead"

"Ruby is Less Popular"

TIOBE Index

11th out of 150 languages

Redmonk Index

8 out of 20 languages

Ruby is Good Enough

Big Sites Use Ruby

- Github
- Airbnb
- Instacart
- Cookpad

Ruby is Fast Enough

For Most of the Cases

If You Reach The Limit, It's OK

Twitter Story

Twitter

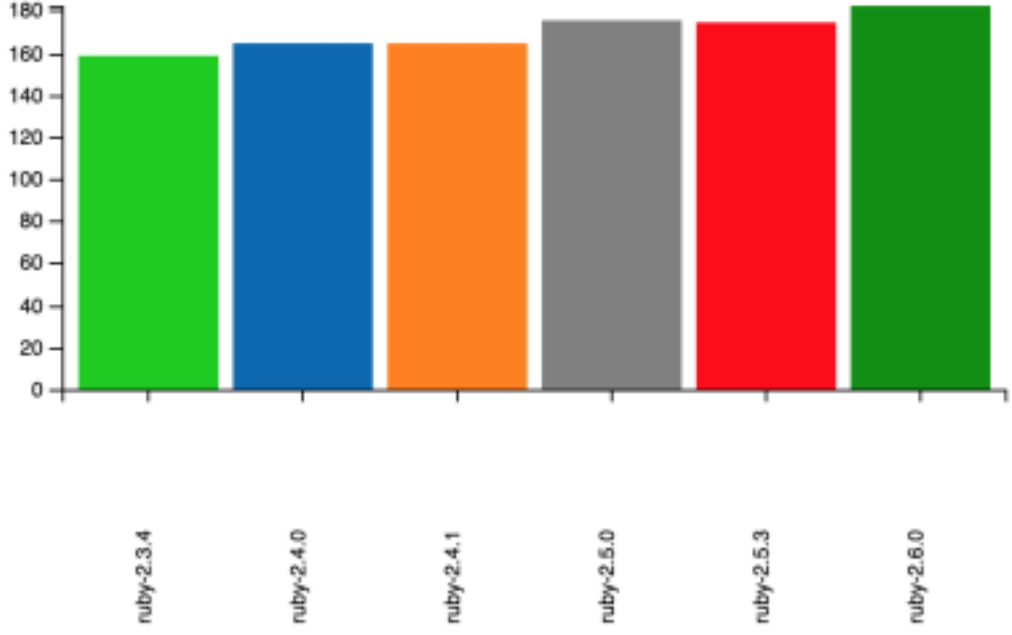
Ruby1.8

Ruby1.9+ is Far Faster

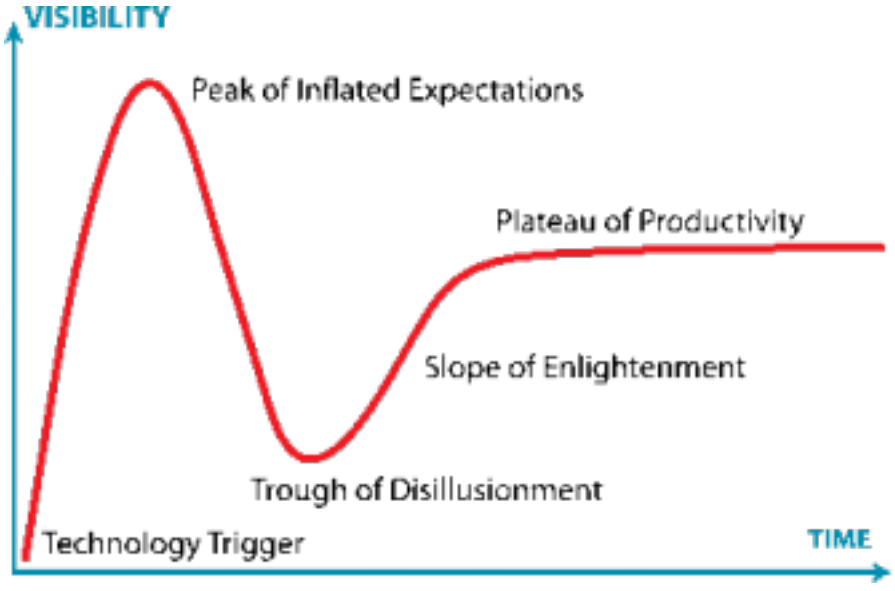
And The Limit is Moving

Ruby2 Improves Performance

Rails Ruby Bench Throughput by Ruby Version



Hype Cycle


VISIBILITY
Peak of Inflated Expectations
Plateau of Productivity
Slope of Enlightenment
Trough of Disillusionment
Technology Trigger
TIME

Let's Face It

Ruby is not Perfect

Some Issues

- Performance
- Multi-Cores
- Bigger Team/Project

Performance

Multi-Cores

Bigger Team/Project

OSS community cannot stop

We Keep Improvement

Even Further in Ruby3

Ruby3 the Future

But How We Can Improve?

- Performance
- Concurrency
- Static Analysis

Static Analysis

Static Typing

As the Project Grows

Test Becomes a Burden

Test Increases Its Size

Test Execution Takes More Time

Tests are Not DRY

Don't' Repeat Yourself

We Do Write Tests Anyway

But Want to be More Productive

What Other Languages Try

PHP: Type Hinting

Python3: Type Annotation

JavaScript: TypeScript

What Should We Do?

Adding Type Annotation?

Like PHP or Python?

No

I Hate Type Annotations

It's Not DRY

We Won't Add Type Hinting

Because It's Not Needed

The Plan for Ruby3

Static Type Checking Components

- Type Definition Syntax (.rbs)
- Type Definitions for Libraries
- Type Profiler
- Static Type Checker

Type Definition Syntax

Soutaro Matsumoto

This is Compromise

Separated Files to describe Types

- Argument Types
- Return Value Type
- Class / Module (Generic)
- Interface

```
class Foo
  def foo: -> void
  def to_s: -> String
          |(Integer) -> String
end
```

A Tool to Parse rbs File

github.com/soutaro/ruby-signature

Type Definitions for Libraries

Standard Libraries

Gems (TBD)

Type Profiler

Yusuke Endoh

The Key Component of Static Analysis

Abstract Interpretation

```
def foo(a)
  a + 2
end
foo(15)
```

```ruby
def foo(a)   # a is int
  a + 2      # int has '+': OK
end
foo(15)      # foo is called with int
```

Type Profiler

- Collect Type Information
- Detect Type Conflict of Type Information
- Generate rbs Files

Type Profiler

- Level 1 Type Checker
- rbs Generator

You can Refine Generated rbs Files

We May Also Provide the YARD to rbs Converter

Static Type Checker

- Sorbet from Stripe
- Steep

Use rbs Type Definitions

Different Characteristic

Sorbet

github.com/sorbet/sorbet

Sorbet

- Fast (in C++)
- (Mostly) Nominal
- Supports type annotation DSL
- rbs Support is Coming

Steep

github.com/soutaro/steep

Steep

- Written in Ruby
- Structural Typing
- Flexible

Healthy Competition

What will Happen with These?

Your ordinary Ruby programs will be statically type checked

Without any type annotations in your Ruby programs

If you refine type definition files,
you will have better checking

Wouldn't That Be Cool?

We Are Working on It

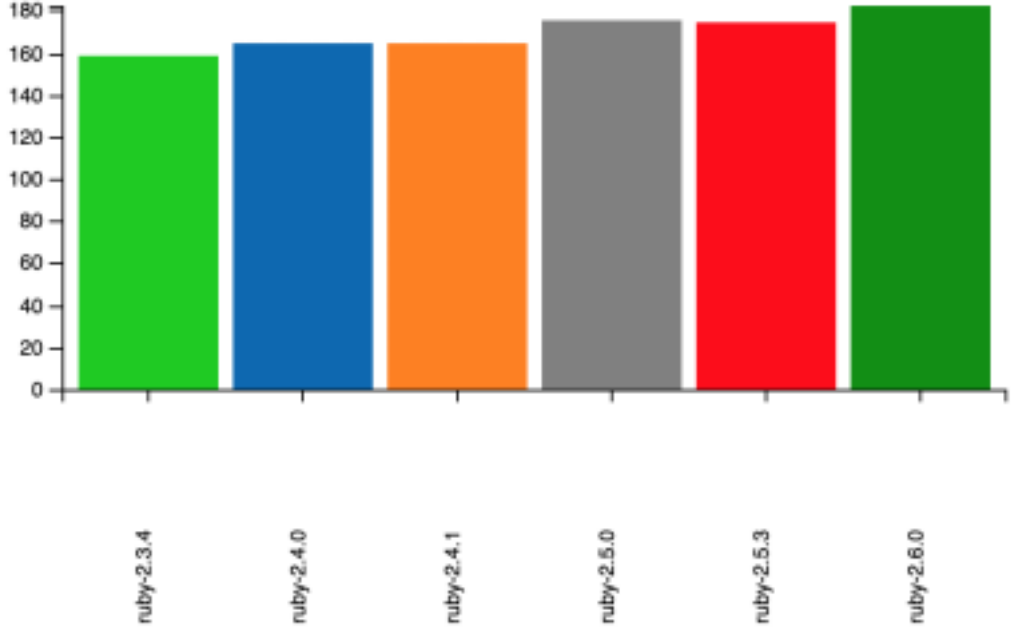
And Result is Promising (at least from my POV)

Performance

No Language can be Fast Enough

Ruby2 Improves Performance

Rails Ruby Bench Throughput by Ruby Version



Even Further in Ruby3

We Need More Performance

Bigger Services

Anxiety

We need to address bottlenecks

- Memory
- CPU
- I/O

Memory is the First Bottleneck

GC Improvement

- Generational GC (2.1)
- Incremental GC (2.2)
- Transient Heap (2.6)
- Object Compaction (2.7?)

Resolve CPU Bottleneck

JIT

Just-in-Time Compiler

MJIT (2.6)

For CPU Intensive Task

`optcarrot` (NES emulator)

Ruby2.6 Runs 2.8x Faster than Ruby2.0

Ruby2.7 Runs Even Faster

For Rails apps

MJIT Runs **Slower**

- Memory Bottleneck
- Too Many Methods
- I/O Intensive

Lighter Compilation

MIR: the lightweight JIT (by Vlad)

3 tier JIT

VM→MIR→MJIT

JIT may not be useful for Web Apps

JIT would be useful for Research Computing

The Other Way to Improve Performance

Concurrency

Concurrency is Hard

I Regret Adding Threads

- Hard to Use Correctly
- Hard to Use Efficiently
- Hard to Debug

We Need Better Abstraction

- Guilds (Isolates)
- AutoFiber (AsyncWhatever)

We Need Better Abstraction

Guilds for CPU Bottlenecks

AutoFiber for I/O Bottlenecks

Easy to Use

Easy to Debug

Easy to Perform

Go or Elixir use Single Entity

We Need Better Names

- Guilds (or Isolate?)
- AutoFibers (or just change Threads?)

Your Opinions are Welcome

Improvements Are Inspired by Functional Programming Languages

Static Typing

Concurrency Models

And Even More

Numbered Block Parameters

From Scala, Clojure, Groovy

```
[1,2,3].map{_1 * 2}
```

Other Options

- it (Groovy, Kotlin)
- @, @1, @2..
- %0, %1, %2..@

Pattern Matching

Not Regular Expression

From Many FP Languages

```
case JSON.parse(json, symbolize_names: true)
in {name: "Alice", children: [name: "Bob", age: age]}
  p age
in _
  p "no Alice"
end
```

```
if person[:name] == "Alice"
  children = person[:children]
  if children.length == 1 && children[0][:name] == "Bob"
    p children[0][:age]
  end
end
```

Chaining (Pipeline) Operator

From F#, Elixir

```
1..100
  |> map{|x| rand(x)}
  |> sort
  |> reverse
  |> take 5
  |> display
```

```
# instead of
(1..100)
  .map{|x| rand(x)}
  .sort
  .reverse
  .take(5)
  .display
```

Pipeline Operator in F# (and ML)

- let (|>) x f = f x
- add primary argument at the **last**

Pipeline Operator in Elixir

- macro
- add primary argument as the **first** argument

The Concept of Pipeline

Add Primary Argument to the Call

Pipeline Operator in Ruby

Add Primary Argument to the Call

Add Primary Argument as the Receiver

- alternative syntax
- different operator precedence (like blocks)
- less parentheses

I gave up

Note that it's **before** 2.7

I had to experiment

At least, we get some ideas

- Allow comments in method chains
- Right assignment => (?)

```
# instead of
(1..100)
  .map{|x| rand(x)}
  .sort
  #.reverse
  .take(5)
  .display
```

Comments in method chains

```
# instead of
(1..100)
  .map{|x| rand(x)}
  .sort
  .reverse
  .take(5) => top_five
```

Right assignment

We Need to Survive

To Provide Benefit to Our Users

To Sustain Our Lives

We Will Keep Moving Forward

With You

To Make You (and Us) Happy

To Make the World Better Place

Thank you