Ruby3 and Beyond

Ruby Association

Yukihiro "Matz" Matsumoto
@yukihiro_matz

Ruby3

Will be Available on December 25th, 2020

As Planned

Unless Something Extremely Bad Happens

It Should be Compatible

It Should be Faster

It Should be Better

Ruby3 is the Future

We Should Keep Compatibility

We Try to Keep Full Compatibility

Except for a few Bug Fixes

Some Program Can Rely on Buggy Behavior

New Features Done Keeping Compatibility

Ruby3.0 Should be Faster

Performance

Performance Matters

Ruby1.9 Tragedy

5+ Years Community Split

Similar Tragedy happens in Python

Python3 Problem

15+ Years to Migrate to Python3

5 years to 15 years

What's the Difference?

Ruby1.9 had YARV

Far Faster VM than 1.8

Performance Matters

Ruby is Fast Enough

Everyone Loves GitHub

Everyone Loves Shopify

They Run on Ruby

No One Cares

They Say "Ruby is Dead"

While They use GitHub

While They buy things on Shopify

They Trust Micro-benchmarks

Reputation is Formed by Micro-benchmarks

Sigh

OK

JIT

Just-In-Time Compiler

Dynamically Generates Native Code

MJIT (Since Ruby2.6)

Ruby3x3

Ruby runs 3 times faster in some benchmarks

Optcarrot

But Not in Rails applications (yet)

We are Improving

with Micro-benchmarks Too

More VM Optimization to Come

Concurrency

1993

One CPU per a Machine

Buy New Computer,
Your Software Run Faster

No More Free Lunch

Single Core Performance Saturation

Multi-core Age

Dual Core, Octa Core, 64 Core

Use More Cores,
Your Software Run Faster

Concurrency is a Key

GIL

Thread-safe Runtime is Hard

Especially for Existing Languages

Our Ideas are:

- Async Fiber
- Ractor

Async Fiber

For I/O heavy Tasks

Async, Await in Other Languages

e.g. node.js

Every I/O with callbacks

I/O with promises

I/O with async, await

Ruby2

Blocking I/O

Ruby3

Async I/O with Fibers

We use Fibers

No Keywords

Context Switch on I/O operations

You need to specify a Scheduler

Async Fiber does not use Multi-Core

How can we use Multi-Core?

For CPU Intensive Tasks

Ractor

Ruby Actor

For CPU intensive Tasks

Isolated Object Space

Communication via (sort of) Channels

No State Sharing

Limited Sharing

- Immutable Objects (Numbers, Symbols,...)
- Deeply Frozen Objects
- Class / Module

No GIL between Ractors

Utilize Multi-Cores

Ruby3.0 Should be Better

Error/Bug Finding

Age of Static Typing

- Go
- Rust
- Swift

- PHP
- Python
- TypeScript (JS)

Everyone Goes Static Typing

Shall We?

NO!

We Want More Precise Checks

We Don't Want Declarations

We Want to Detect Error Earlier

Static Type Checks

- Ruby Signatures
- TypeProf

Ruby Signatures (RBS)

TypeScript d.ts Counterpart

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

Ruby3 ships with RBS for the core

Type Checker

Better IDE

Better Code Completion

Type Signature Pop-up

TypeProf

Naive Type Checks

RBS Generation (for your app)

```
class Foo
  def foo(a)     # a is int
    b = a + 2    # int has '+': OK; b is int
    return b     # returns int
  end
  def bar(a)     # a is int
    b = a + "2"  # int has no '+' with String: error!
    return b
  end
end
Foo.new.foo(15) # foo is called with int
f = Foo.new
f.foo(15)
f.bar(42)
```

Abstract Interpretation

```
class Foo
  def foo:(Integer) -> Integer
end
```

We Want Even Better Ruby

New Syntax

- Pattern Matching
- Numbered Block Parameters

Pattern Matching (2.7)

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

Numbered Block Parameters

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

Numbered Block Parameters

And Beyond...

Seek for Completion

Some New Features May be Buggy

Especially Ractors

Need Improvement

Supporting Tools

Solargraph

Sorbet

Rubocop

Better Tools Enables Better UX

We Need More Tools

- Type Checker
- Formatter
- Language Server Protocol
- Performance Tuning

Faster

Better JIT

Multi-Layer JIT

Lightweight JIT

MIR (or DynASM)

One More Thing...

Static Ruby

Ruby the Dynamic

- Dynamic Typing
- Dynamic Class (Open Class)
- Metaprogramming

- Useful
- Flexible
- Powerful

Hard to Optimize

Static Barrier

Some Dynamic Aspect Will be Prohibited
Beyond This Barrier

e.g. Method Redefinition

Method Cache Will Work More Efficiently

Static Interpretation

Or (Limited) Macro

Reform AST

It's Just an Idea

Anyway,

We Will Keep Moving Forward

To Create Better World

And You Too

Thank you