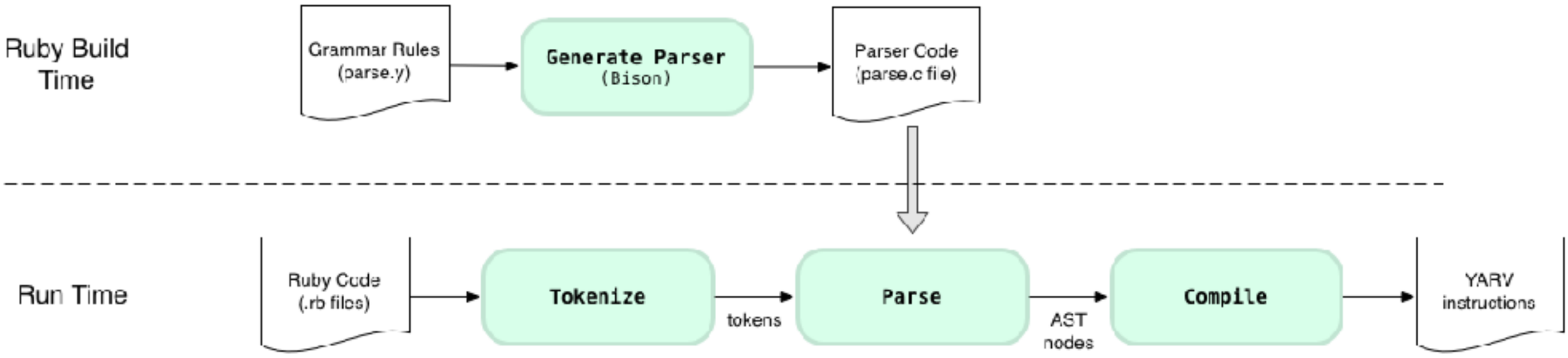
Ruby's JIT

by @jimmynguyc

Hi

- Jimmy Ngu
- Engineering Team Lead
- RapidRiver Software
- KL Ruby Brigade, RubyConf MY
- I present a lot of useless talks :D

How Ruby runs your program?



Tokenize

```
# This is a string
"x > 1"

# These are tokens
["x", ">", "1"]
```

```ruby
require 'ripper'
Ripper.tokenize("x > 1 ? 'foo' : 'bar'")
# => ["x", " ", ">", " ", "1", " ", "?", " ", "'", "foo", "'", " ", ":", " ", "'", "bar", "'"]
```

Parse

```
require 'ripper'
require 'pp'

pp Ripper.sexp("x > 100 ? 'foo' : 'bar'")

# [:program,
#  [[:ifop,
#    [:binary, [:vcall, [:@ident, "x", [1, 0]]], :>, [:@int, "100", [1, 4]]],
#    [:string_literal, [:string_content, [:@tstring_content, "foo", [1, 11]]]],
#    [:string_literal, [:string_content, [:@tstring_content, "foobar", [1, 19]]]]]]]
```

S-expressions

Abstract Syntax Tree (AST)

S-expressions

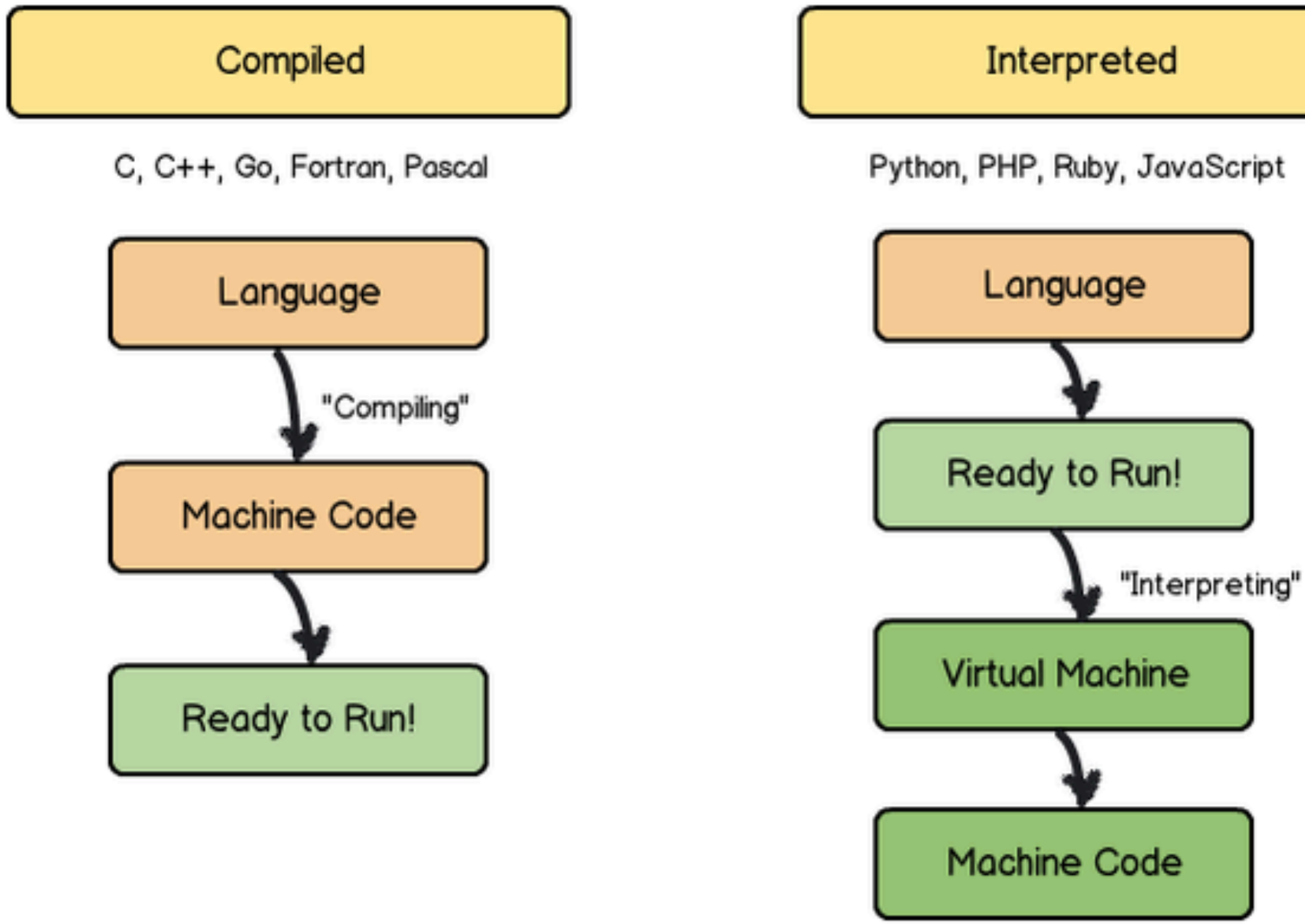
```
# Define a progam
[:program,
 # Do an "if" operation
 [[:ifop,
   # Check the conditional (x > 100)
   [:binary, [:vcall, [:@ident, "x", [1, 0]]], :>, [:@int, "100", [1, 4]]],
   # If true, return "foo"
   [:string_literal, [:string_content, [:@tstring_content, "foo", [1, 11]]]],
   # If false, return "bar"
   [:string_literal, [:string_content, [:@tstring_content, "foobar", [1, 19]]]]]]]
```

Compile

```
puts RubyVM::InstructionSequence.compile("x > 100 ? 'foo' : 'bar'").disassemble
# == disasm: <RubyVM::InstructionSequence:<compiled>@<compiled>>==========
# 0000 trace            1                                               (   1)
# 0002 putself
# 0003 opt_send_without_block <callinfo!mid:x, argc:0, FCALL|VCALL|ARGS_SIMPLE>
# 0005 putobject        100
# 0007 opt_gt           <callinfo!mid:>, argc:1, ARGS_SIMPLE>
# 0009 branchunless     15
# 0011 putstring        "foo"
# 0013 leave
# 0014 pop
# 0015 putstring        "bar"
# 0017 leave
```

Bytecodes

Compiler vs Interpreter



What is JIT

- Just-In-Time Compiler

- “Smartly” Converts part of code most used into “Machine Language”

Ruby's JIT

- JIT is not new to Ruby !!
- JRuby (JVM), Rubinius (no longer), TruffleRuby (GraalVM), Topaz (RPython)
- Previous attempts - rujit, llrb (using LLVM's JIT Library)
- Main issue - relying on external JIT projects

MJIT

- Not until 2016
- MJIT by Vladimir Makarov
- Brand new Compiler & VM Instruction definitions
- Register based instructions using Register Transfer Language (RTL)
- Problem - require a lot of work & test to be stable
- At least another year or 2 to be stable

YARV-MJIT

- 3 months ago by k0kubun (Takeshi Kokubun)
- Creator of llrb project
- Register based VM, no change to VM instructions
- Doesn’t change existing Ruby program behaviors
- Conservative JIT compiler (but still a JIT !!!)
- 80% faster than Ruby 2.0, 30% faster than Ruby 2.5 on optcarrot
- Releasing in Ruby 2.6

References

- https://bugs.ruby-lang.org/issues/12589
- https://bugs.ruby-lang.org/issues/14235
- http://engineering.appfolio.com/appfolio-engineering/2017/12/26/ruby-3-and-jit-where-when-and-how-fast
- https://medium.com/square-corner-blog/rubys-new-jit-91a5c864dd10
- https://medium.com/@k0kubun/the-method-jit-compiler-for-ruby-2-6-388ee0989c13
- http://blog.honeybadger.io/how-ruby-interprets-and-runs-your-programs/

End