Origins of Ruby & Rubinius

Ruby, the way it meant to be

Origins of Ruby

- Smalltalk
- Perl
- Lisp
- CLU

Origins of Ruby

- Object model from Smalltalk
- Syntax from Perl
- Built-in regexps from Perl
- Metaprogramming from Lisp
- Lambdas from Lisp
- Blocks from CLU

Smalltalk

- One of first OO languages
- Very fast implementations
- Programmable runtime
- Fairly rich standard library
- Compiler

Perl

- Extremely flexible syntax
- Insanely complex parser and lexer
- Built-in regular expressions
- String interpolations

Lisp

- Very dynamic
- Code is data
- Metaprogramming
- Lambdas, functions are first class citizen
- predicates? and bang! methods
- nil
- Fast VMs

Rubinius

- Ruby, the Smalltalk way
- Well, kind of
- With ideas from other languages
- RubySpec

Ruby, the Smalltalk way

- Advanced VM
- Generational GC
- Object memory
- Method inlining and caching
- (At least to some extent) programmable runtime
- Foreign function interface

Object model

- Everything is an object
- I mean, REALLY
- Object allocation
- Garbage collection
- Functions/calls stack

ObjectMemory

- Performs object allocation
- Aggregates Garbage Collector
- Available to application code via special API

GC

- Generational
- Actually 2 (3?) GCs for different purposes
- Immix GC (very efficient, paper recently published)
- Actually deallocates memory (MRI does not)

Compiler

- Written in Ruby
- 3rd (or something) rewrite
- Plugin architecture

Virtual machine

- Bytecode VM
- Just-in-time compilation
- Based on LLVM IR
- Supports old C extensions (via shim C API)
- Built-in classes implementation is OO (C++) and maps Ruby OO model very well

Multiple VMs

- Actors model (a la Erlang, Scala)
- background_runner = VMActor.new(...)
- background_runner.dispatch(:operation, ...)

When?

- When it’s done

If you have other questions...

- Jabber: michael@novemberain.com
- Skype: michaelklishin

Thank you