Compiling Ruby scripts

Koichi Sasada
ko1@heroku.com

Today's talk

- Making ruby script serializer and deserializer
  - Well known technique and tools such as JVM class file
  - No special technique is needed. Just implemented it.
- Introduction of how to use
- Evaluation result

Koichi Sasada

A programmer living in Japan

Koichi is a Programmer

- MRI committer since 2007/01
  - Original YARV developer since 2004/01
    - YARV: Yet Another RubyVM
    - Introduced into Ruby (MRI) 1.9.0 and later
  - Generational/incremental GC for 2.x

Koichi is an Employee

heroku

salesforce heroku

Ayumu Aizawa
Solutions Architect, Heroku

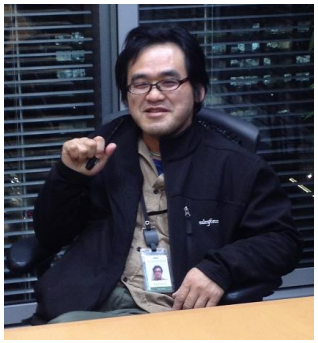
Satoshi Nagano
Heroku sales manager

Koichi is a member of Heroku Matz team

Mission

Design Ruby language and improve quality of MRI

Heroku employs three full time Ruby core developers in Japan named “Matz team”



Upcoming Ruby 2.3

Today, no time to introduce new features…

Please ask me later.

Appreciation

CI server sponsored by YassLab.

YassLab.

Having a Good Life with OpenSource ;)

A CI server for EL Capitann

Setup by Shibata-san

Many CI servers

Ruby CI

Current Reports Latest Reports

Server	Datetime	Branch	Option	Revision	test	test-all	rubyspec	Summary	Diff
Debian 7.5 i686	12-11 09:32	trunk		53027		0F1E		416W 0F1E	diff:test-all,su...
Debian 7.9 x86_64	12-11 06:00	trunk		53027				424W success	diff:src
Debian 8.2 x86_64	12-11 09:30	trunk		53027				419W success	no diff
Debian 9.0(testing) x86_64	12-11 09:30	trunk		53027				414W success	diff:process-s...
Ubuntu 10.04 32-bit	12-11 09:43	trunk		53027		1F0E		413W 1F0E	diff:test-all,su...
Ubuntu 10.04 64-bit	12-11 09:03	trunk		53027				413W success	no diff
Ubuntu 14.04 x86_64	12-11 08:00	trunk		53027				410W success	no diff
Ubuntu 14.04 x86_64n	12-11 10:28	trunk		53027				405W success	diff:test-all
Ubuntu armv7l eabihf	12-11 08:17	trunk		53027				412W success	diff:test-all,su...
Ubuntu 14.04 aarch64	12-11 10:03	trunk		53027		2F0E	0F6E	412W 2F0E rubyspec:0F6E	diff:process-s...
Ubuntu 15.10 x86_64	12-11 09:30	trunk		53027				419W success	diff:process-s...
Gentoo	12-11 03:30	trunk		53026		1F0E		407W 1F0E	diff:src,test-all...
CentOS 5 i386	12-11 09:03	trunk		53027				420W success	diff:test-all,su...
CentOS 5 x86_64	12-11 08:33	trunk		53027				434W success	diff:test-all,su...
CentOS 6 64bit	12-11 10:02	trunk		53027				434W success	no diff
CentOS 7.1 x86_64	12-11 08:00	trunk		53027				420W success	diff:test-all
Fedora 19 ppc64	12-11 09:03	trunk		53027				418W success	no diff
Fedora 21 x86_64	12-11 09:15	trunk		53027				412W success	diff:test-all
Fedora 22 x86_64	12-11 09:15	trunk		53027				421W success	no diff
Amazon Linux 2015.09 x64	12-11 06:30	trunk		53027				413W success	diff:src,proces...
Arch Linux	12-11 09:30	trunk		53027				425W success	diff:src,proces...
openSUSE 13.2	12-11 06:30	trunk		53027				413W success	diff:src
RHEL 7.1 s390x	12-11 09:33	trunk		53027				419W success	diff:src,test-all
FreeBSD 10.1 x86	12-11 10:03	trunk		53027				413W success	diff:test-all
FreeBSD 10.1 x64	12-11 10:03	trunk		53027				413W success	diff:btest,mak...
OS X Yosemite	12-11 08:45	trunk		53027		1F0E		412W 1F0E	diff:test-all,su...
OS X El Capitan	12-11 08:45	trunk		53027		1		412W success	diff:test-all,su...
Solaris 11x	12-11 07:11	trunk		53027				475W success	diff:pflags
Solaris 11s	12-11 06:25	trunk		53027				476W success	diff:pflags,test...
vc10-x64	12-11 10:04	trunk		53027				315W success	no diff
icc16-x64	12-11 06:00	trunk		53027		89F0E	7F0E	3813W 89F0E rubyspec:7F0E	diff:src,miniru...
Solaris 10s	12-11 05:03	trunk		53027		failed		612319W failed(test-all CommandTim...	no diff
Solaris 10x	12-11 08:48	trunk		53027		failed	5F0E	776W 2[BUG] 2[SEGV] failed(test-all) f...	no diff
Ubuntu 16.04(dev) x86_64	12-11 09:30	trunk		53027		1F0E		405W 1F0E	diff:process-s...

So many contributions

Development

- All MRI developers
  - Committers
  - Code and documents contributors
  - Issue reporters
- Heroku
  - Employs full-time MRI committers (Matz, Nobu, Ko1)
- Ruby Association
  - Maintain Ruby 2.1 and 2.0 (security)

Environment

- Ruby Association
  - Sponsored CI servers on Cloud services
- YassLab and N.R.K
  - Sponsored a new mac mini machines for CI server
- Prof. Sugaya, Shibaura Institute of Technology
  - Allows us to locate physical machines
- Travis-CI
  - Provides CI services
- CloudCore, DTI
  - Provides CI servers

Deliver

- Fastly.com
  - Delivers Ruby binaries by their CDN
- NaCl, IIJ, Heroku
  - Host web servers
- GlobalSign
  - Sponsored SSL certification

We need more supports

- Examples
  - **Nobu's development machine**
  - Development/benchmark machines
  - CI machines (VPS)
  - Hackathon travel fee
- See "Misc #11783: Do you have any idea if you have a budgets?"
  - https://bugs.ruby-lang.org/issues/11783

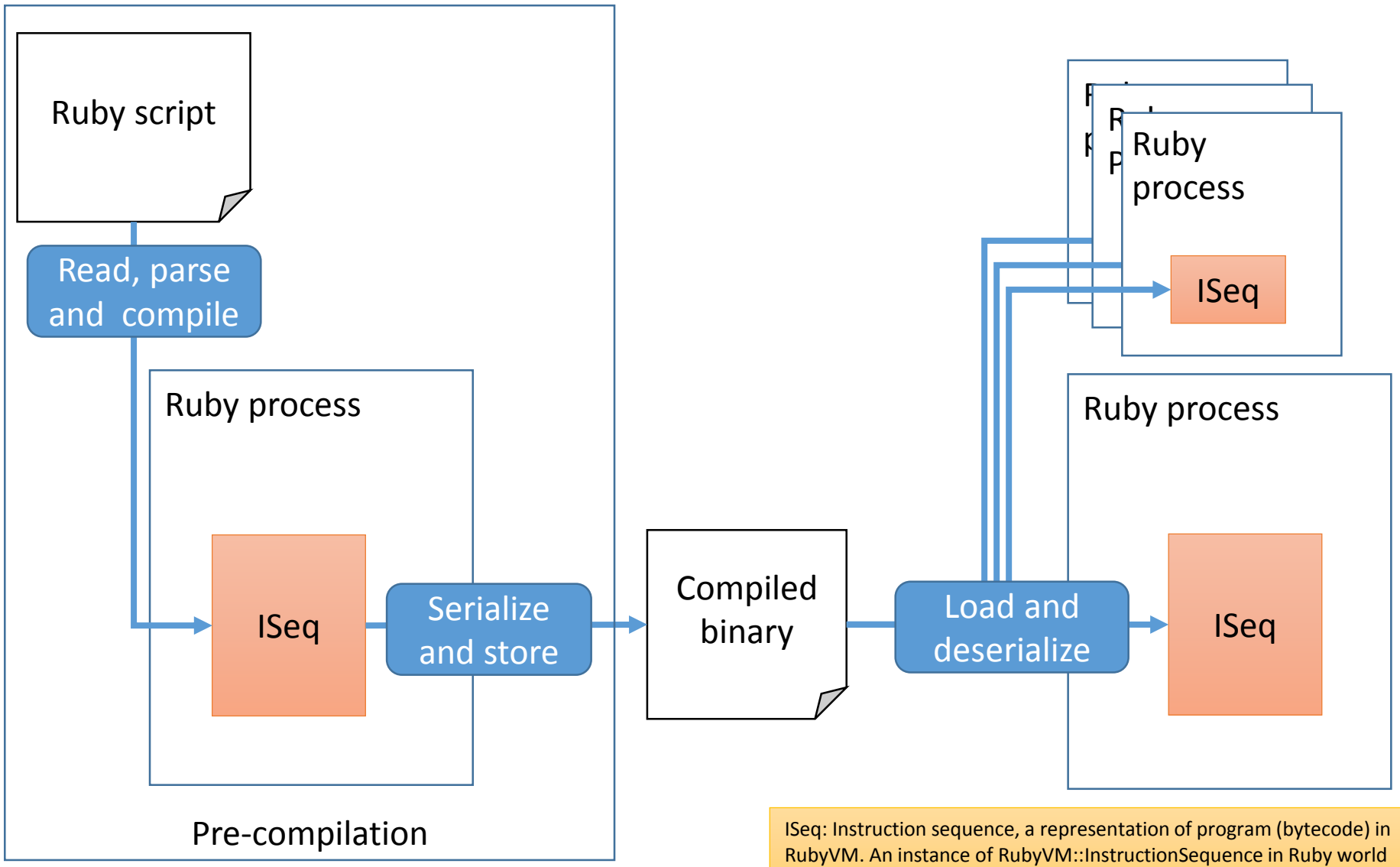
How about to utilize this opportunity for your promotion?

Compiling Ruby scripts

Compilers for interpreters

- JIT compilers
  - Program to native machine code
  - Runtime statistics information are available
- AOT compilers
  - Program to native machine code
  - Program to other languages code
    - Translate to C, Java, etc…
  - Program to persistent byte code
    - RubyVM::InstructionSequence in Ruby's case

Store serialized program and load



ISeq: Instruction sequence, a representation of program (bytecode) in RubyVM. An instance of RubyVM::InstructionSequence in Ruby world

Purpose of ISeq (de)serializer

- Fast boot
- Reduce memory consumption
- Migrate compiled code to other nodes

Purpose of ISeq (de)serializer
Goal of this time

- Fast boot
- Reduce memory consumption
- Migrate compiled code to other nodes

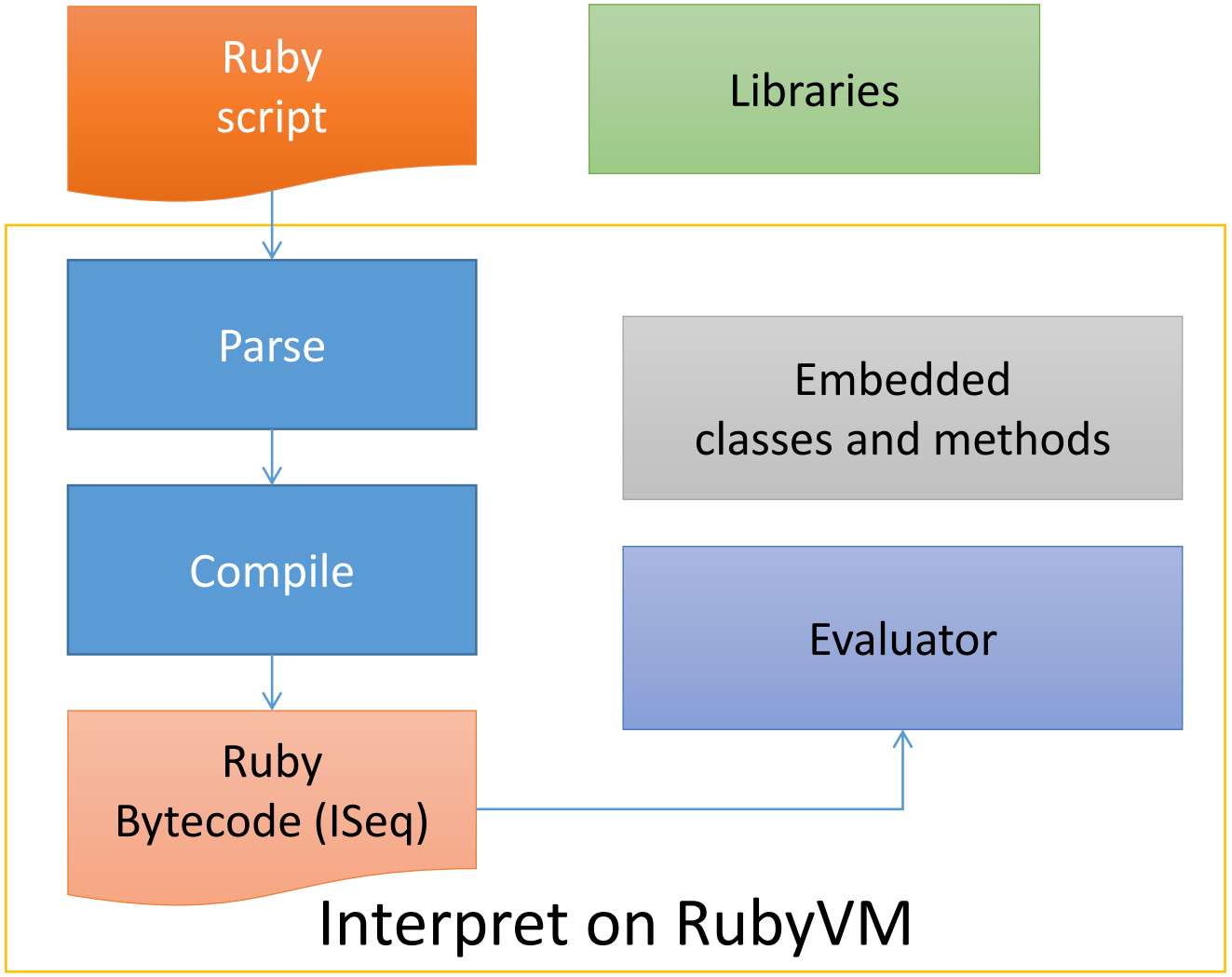
Out of scope

Not support portable binary

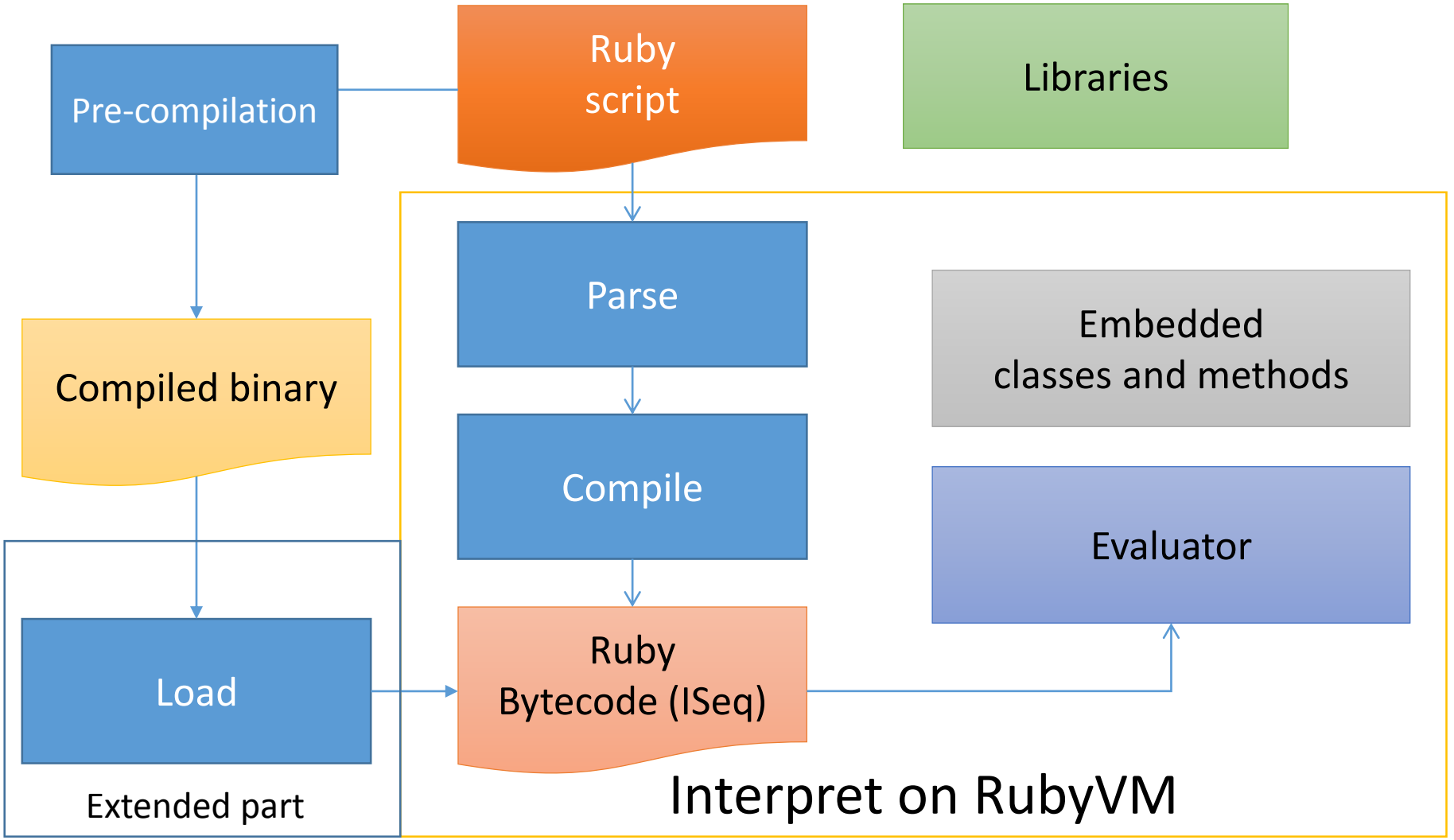
Not verify at loading time

→ Do not believe binaries by others

Fast boot

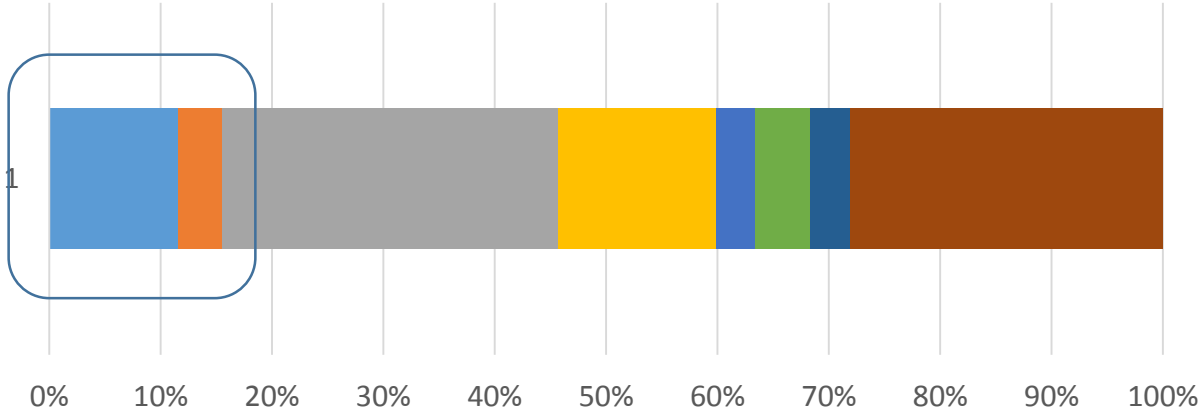

Ruby
script
Libraries
Parse
Embedded
classes and methods
Compile
Evaluator
Ruby
Bytecode (ISeq)
Interpret on RubyVM

Fast boot


Pre-compilation
Ruby script
Libraries
Parse
Embedded classes and methods
Compiled binary
Compile
Evaluator
Load
Ruby Bytecode (ISeq)
Extended part
Interpret on RubyVM

Memory consumption Current issue

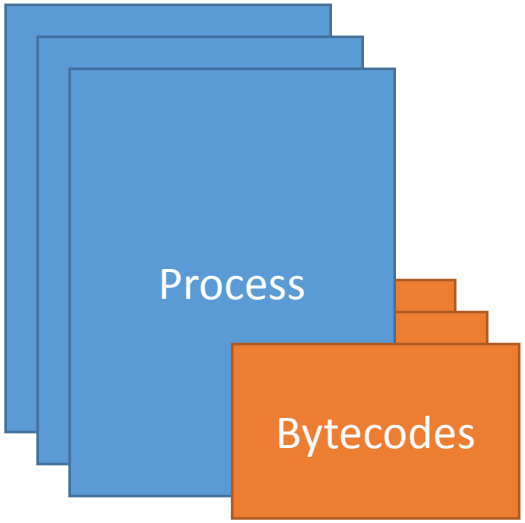
ISeq consumes 15% (20MB) on simple Rails app



	1
iseq_setup@compile.c	15,595,764
rb_iseq_new_with_opt@iseq.c	5,231,136
heap_assign_page@gc.c	40,518,400
st_init_table_with_size@st.c	18,994,480
rb_str_buf_new@string.c	4,817,252
st_update@st.c	6,578,736
onig_region_resize@regexec.c	4,891,968
others	37,676,810

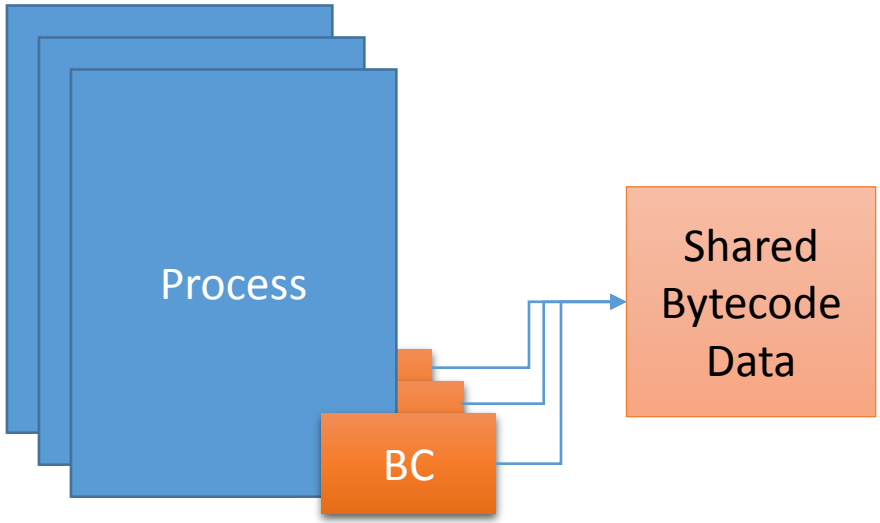
Memory consumption
Current issue on multi-processes

Actual



Independent BCs

Expected

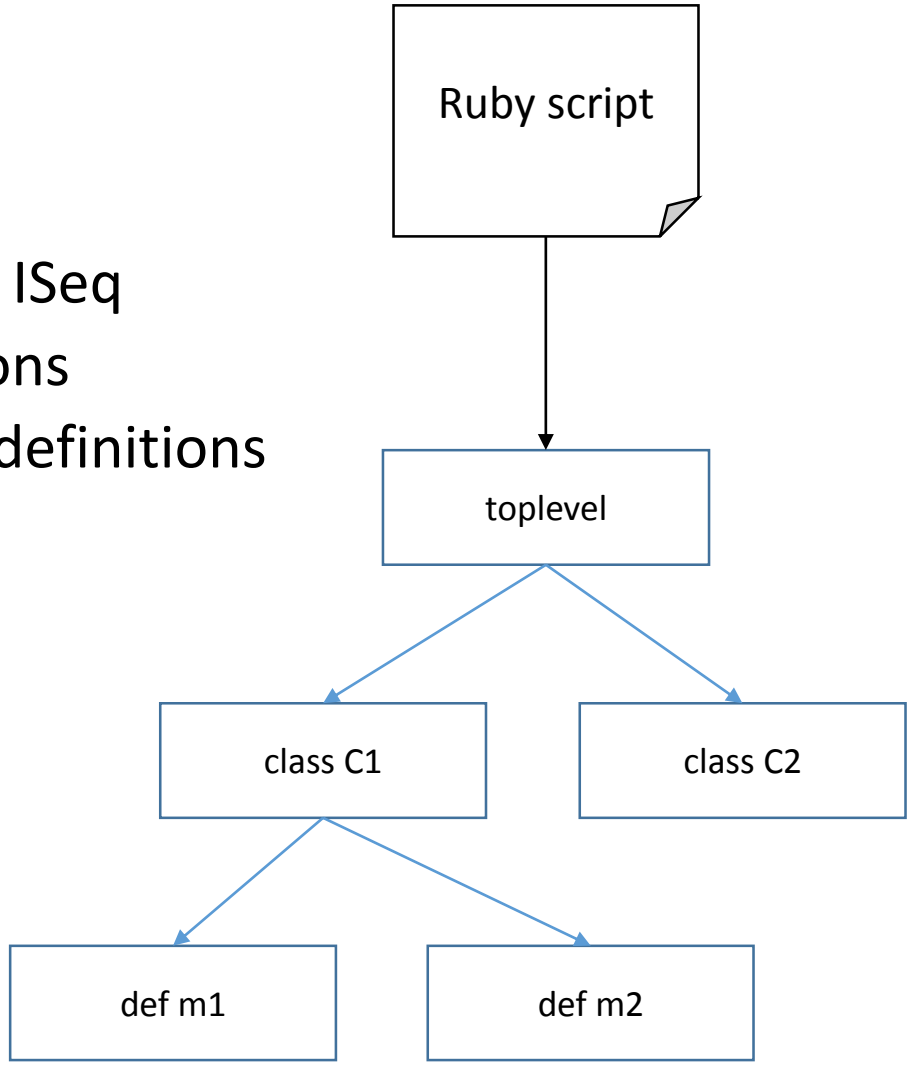


(Partialy) Shared BCs

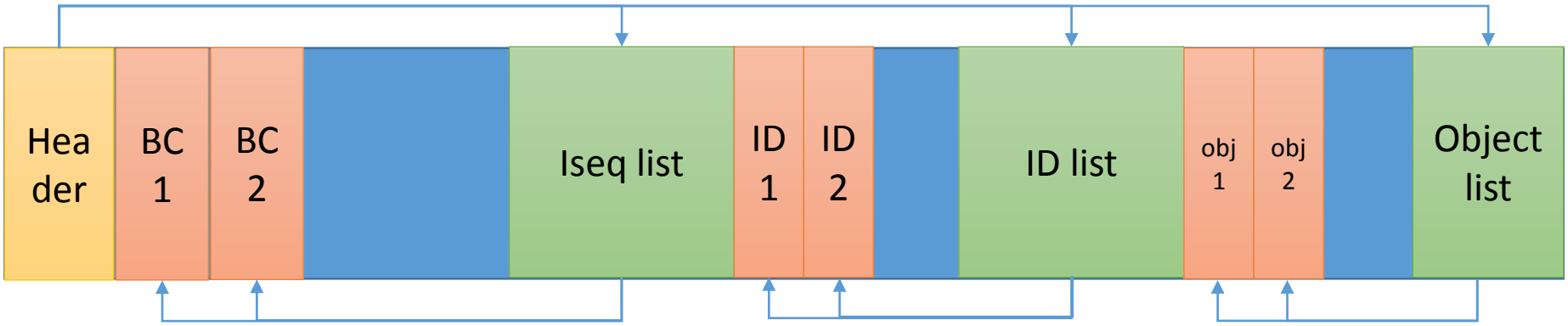
Design and implementation

ISeq tree

- ISeq consists as tree
  - Basically, each scope has own ISeq
  - A top-level has class expressions
  - Class expression has method definitions
  - Method definition has blocks
  - Block has blocks, …
  - Other bytecode blocks
    - ensure, rescue, …
  - And other exceptional cases



Binary data format (not so matured)



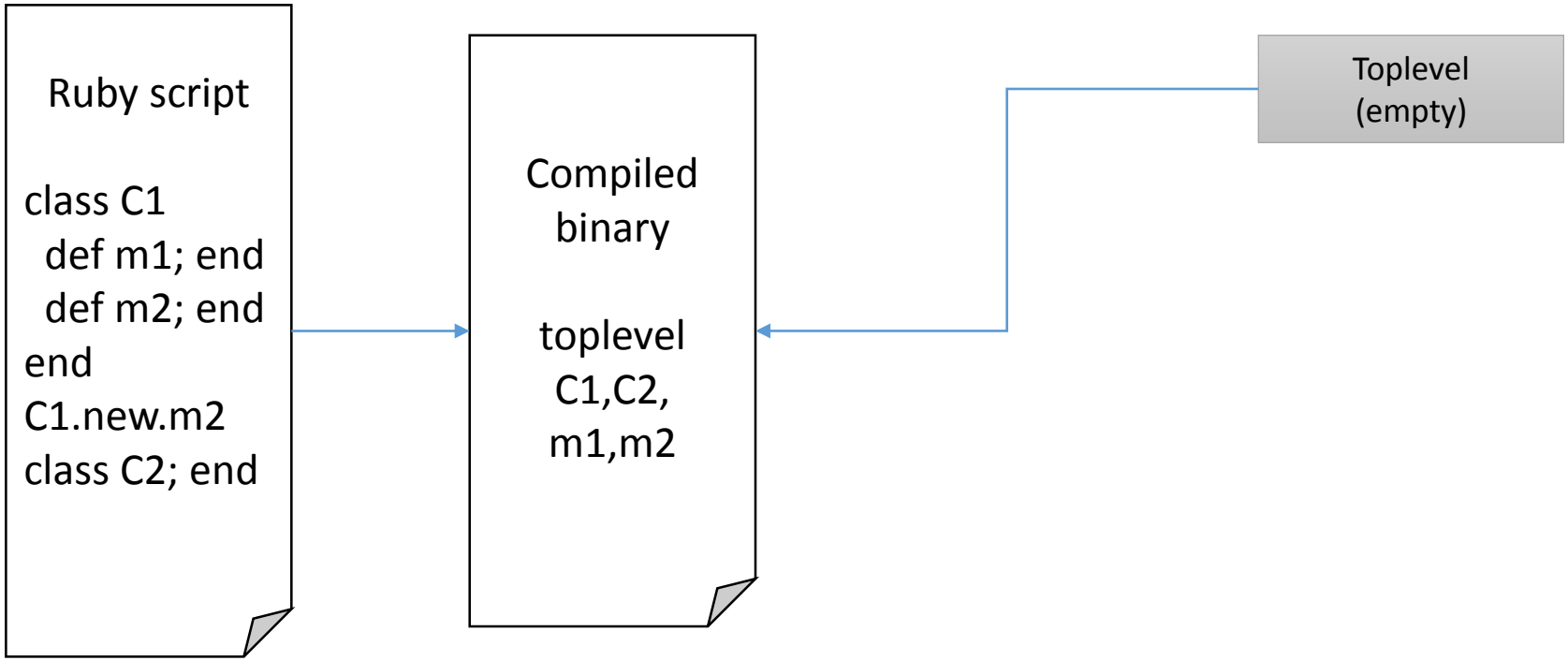
- Iseq (BC), ID, Objects are pointed by index of each lists in each data
- Objects are serialized (manually)
- Dump machine dependent data (can't migrate compiled code)
- No verifier (because this file is not for migrations)

Technique
Lazy loading

- Do not load every bytecode at once
- Load bytecode if needed

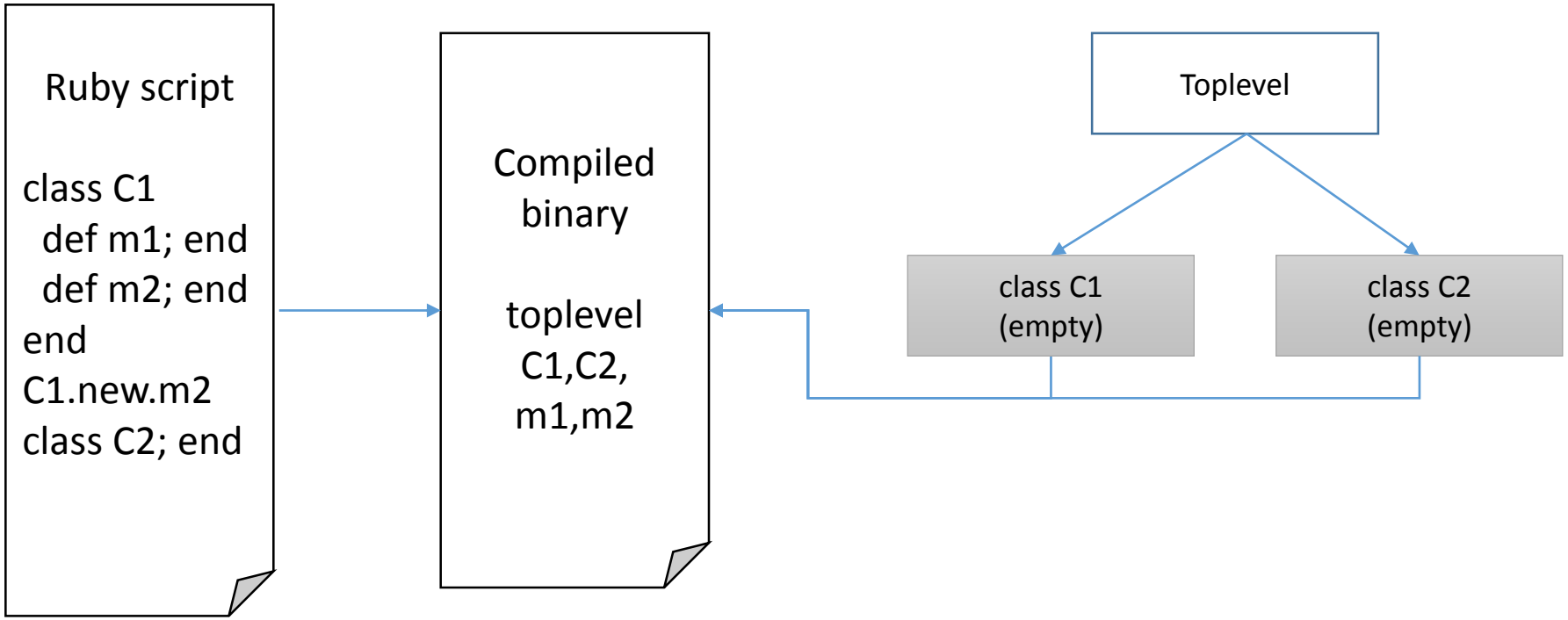
Technique
Lazy loading

(1)Load and make an empty toplevel Iseq



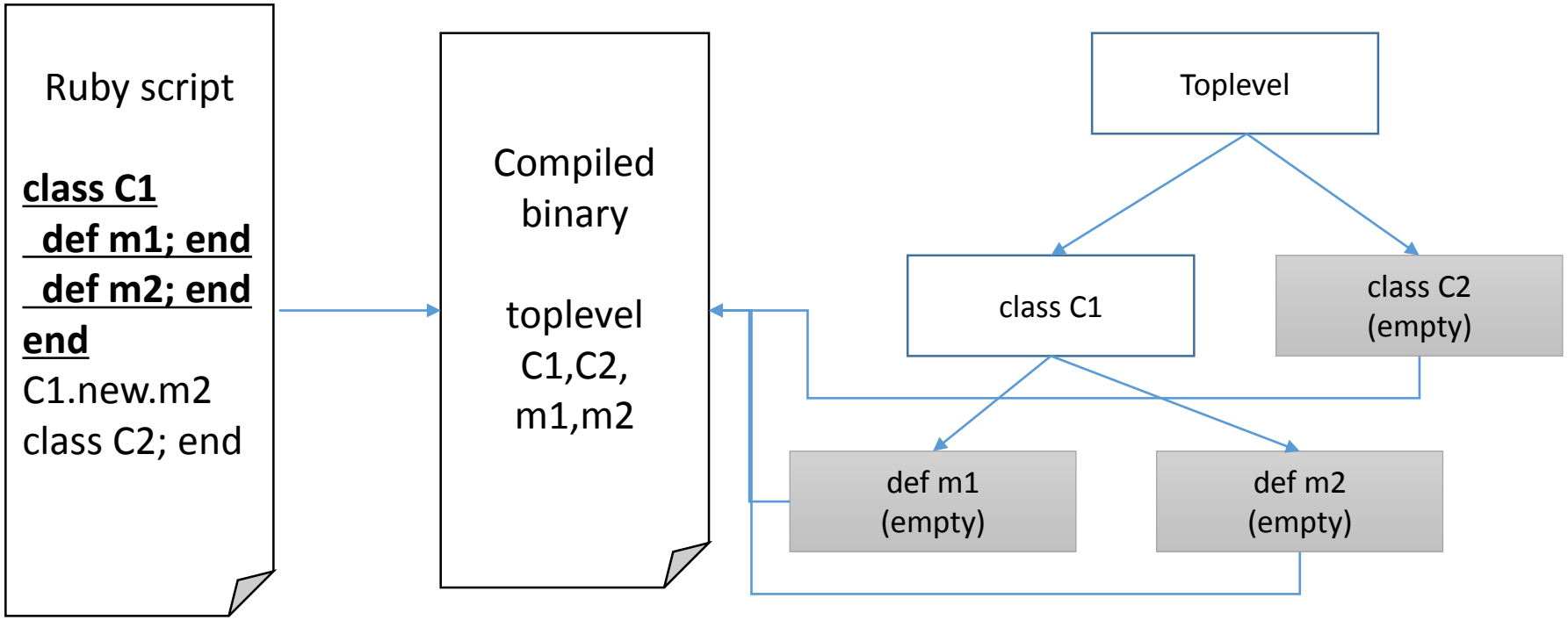
Technique
Lazy loading

(2) Load toplevel ISeq and make empty C1, C2 empty ISeq and evaluate toplevel ISeq



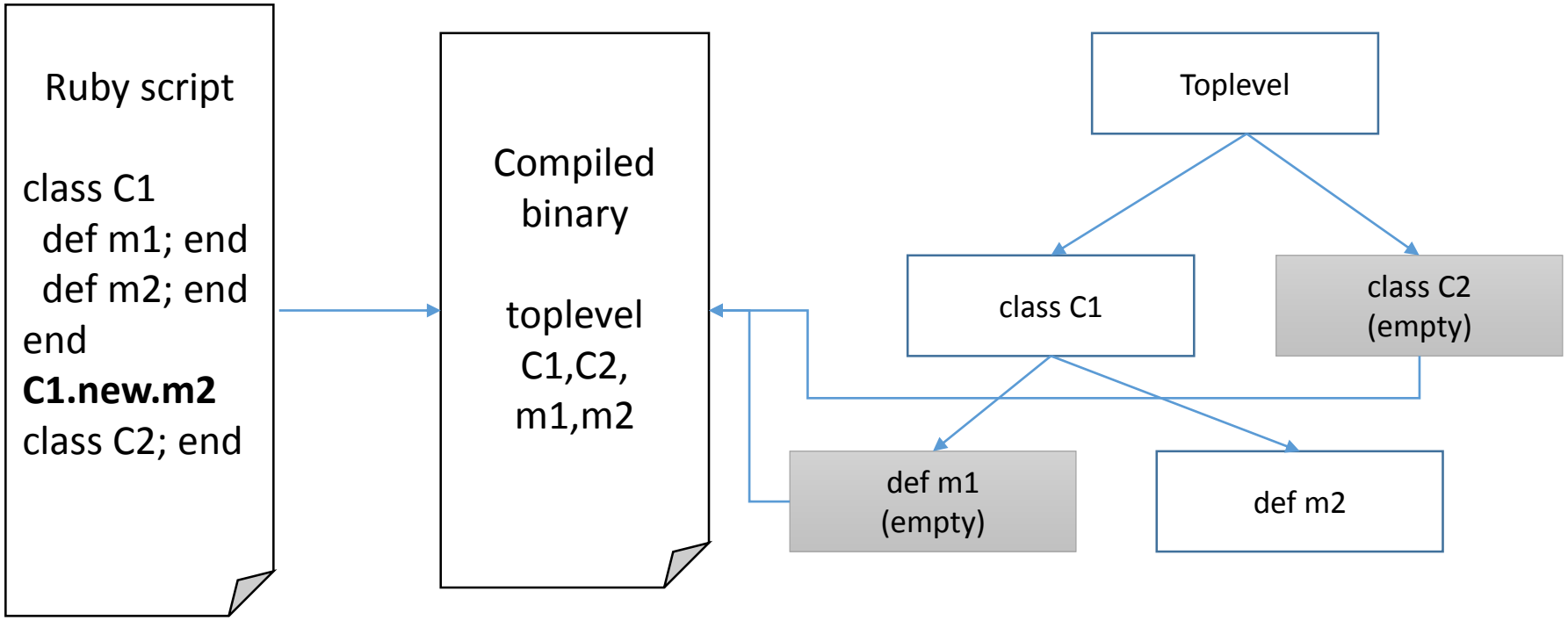
Technique
Lazy loading

(3) Load C1 and evaluate C1
Define m1 and m2 with empty ISeqs



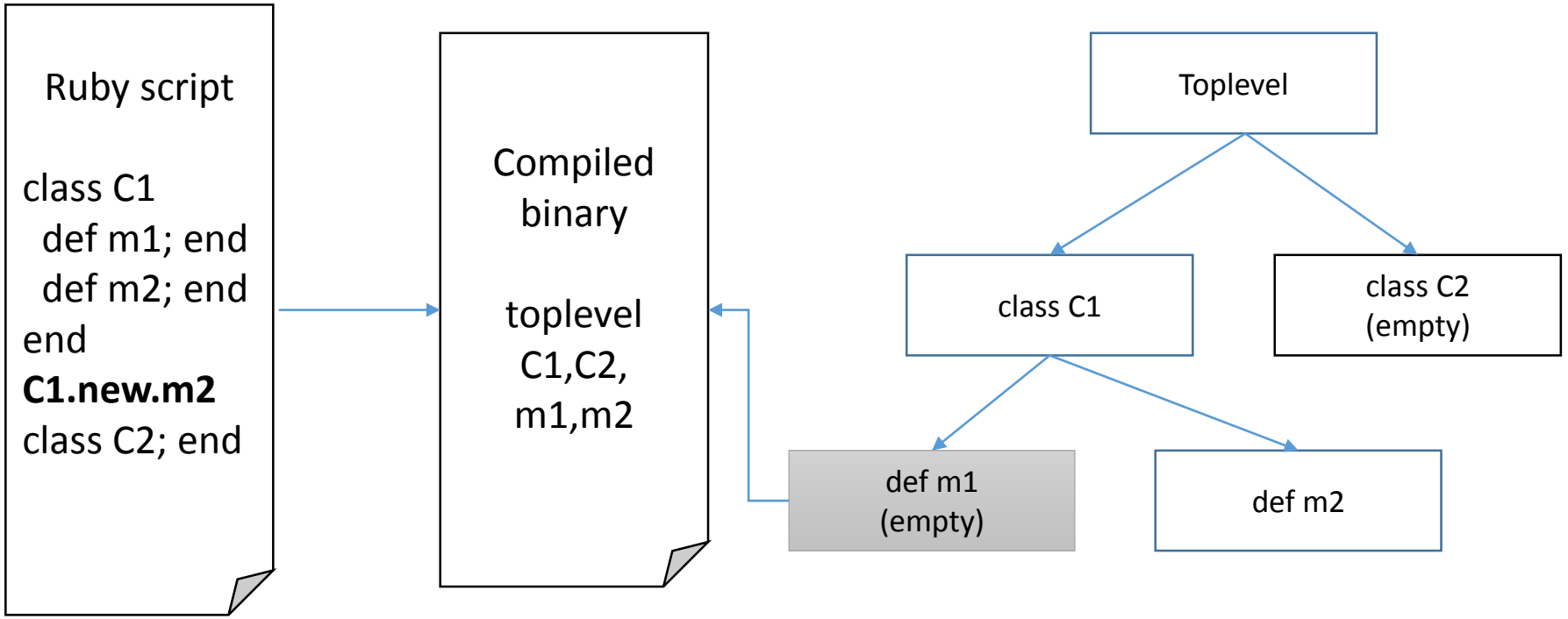
Technique
Lazy loading

(4) Load m2 and invoke m2



Technique
Lazy loading

(4) Load C2 and evaluate C2



Interface
API and Tools

How to store compiled binary?

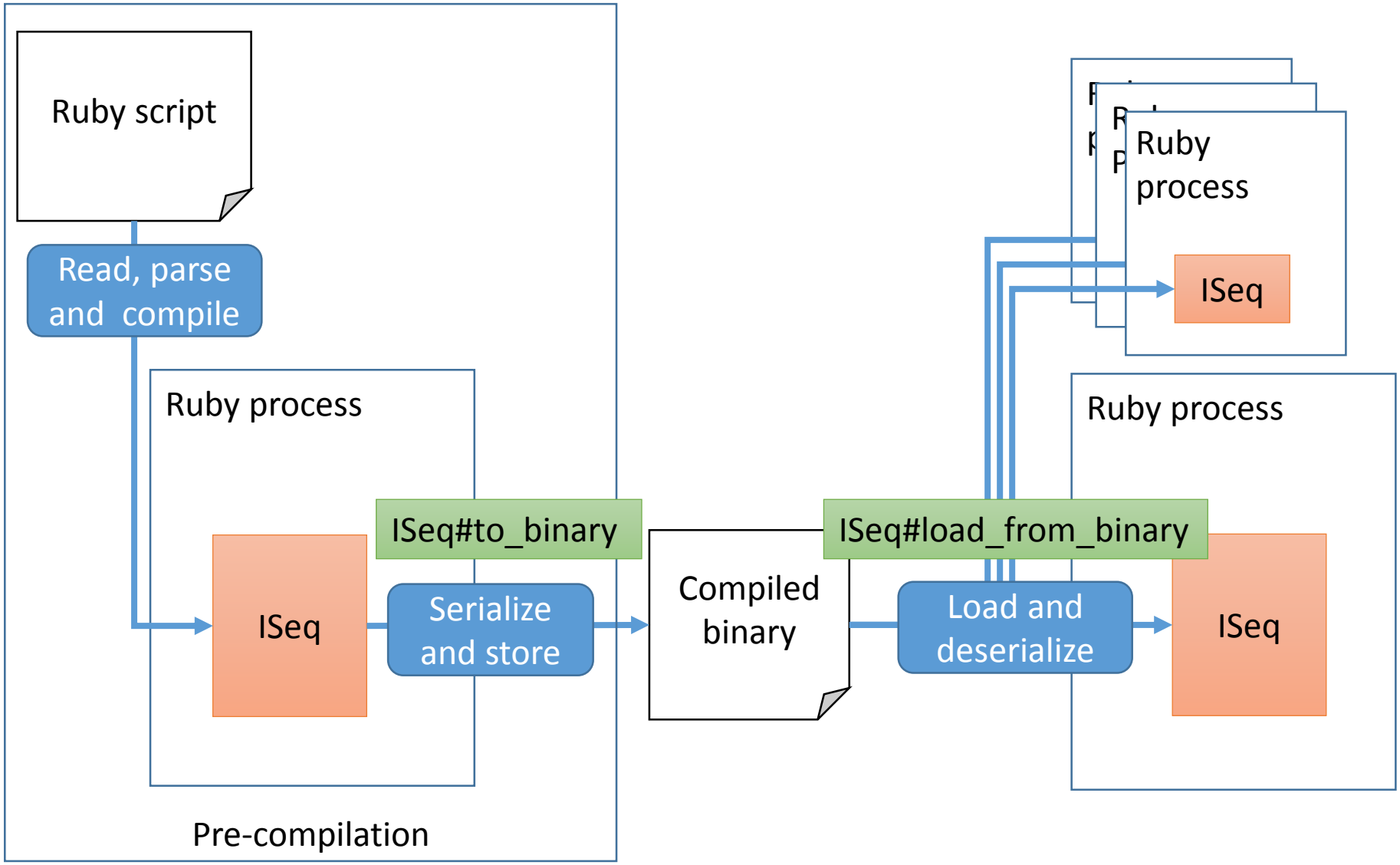
- Compile timing
  - Use compiler explicitly
    - C/Java/… compilers
  - Loading time
    - Rubinius, Python, …
- Location of compiled binary
  - A file in a same directory of *.rb files
  - A file in a special directory
  - DB

Not fixed!

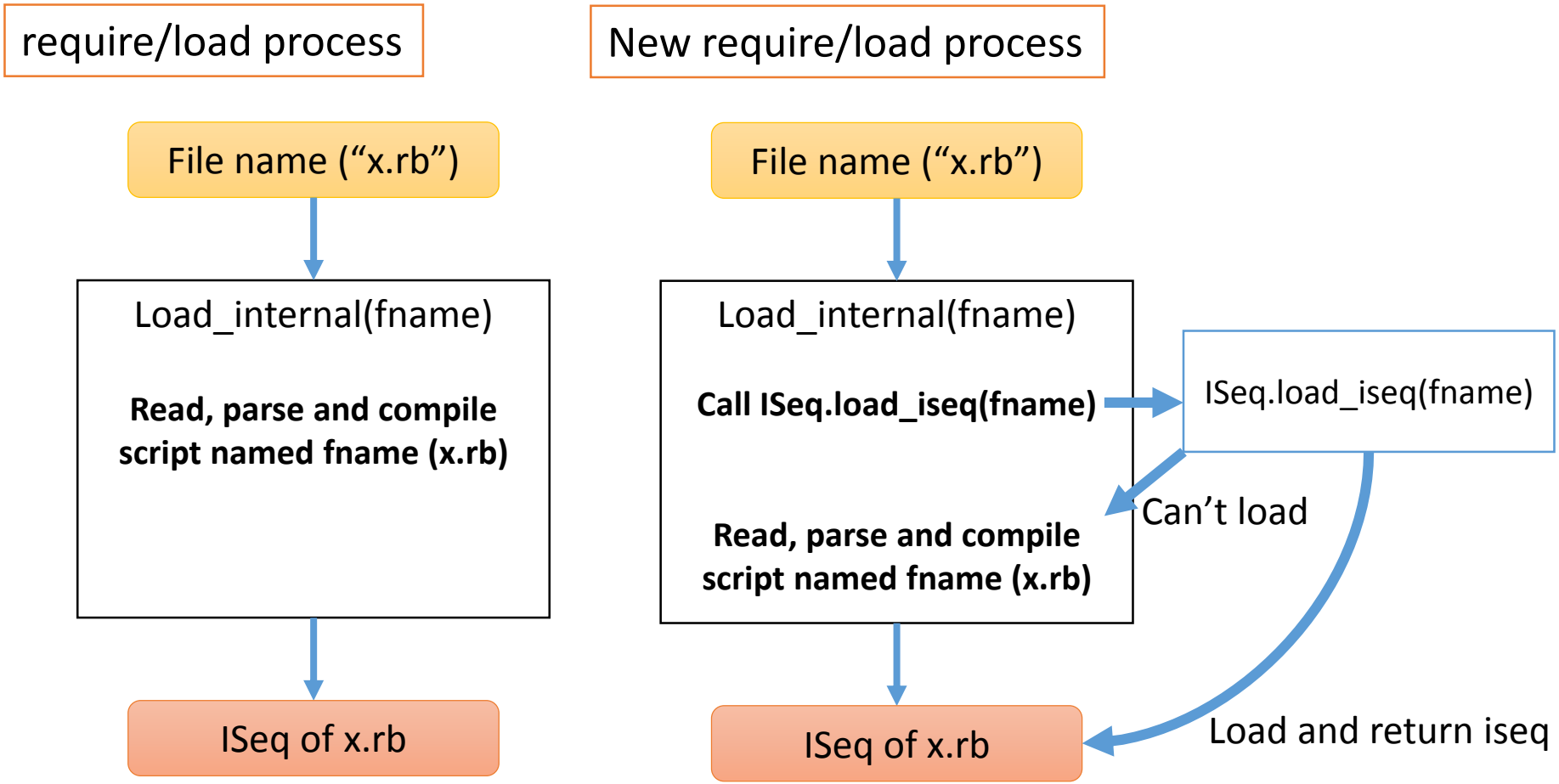
Current implementation Provide loading API

- **RubyVM::InstructionSequence.load_iseq**
  - Call this method at every loading time (if defined)
  - This method should return or loaded ISeq object or nil
  - You can write your own loader
    - From File
    - From any DB
    - From network, and so on
- **RubyVM::InstructionSequence#to_binary**
- **RubyVM::InstructionSequnece.load_from_binary(binary)**
  - Serialize and deserialize (load) methods

Store serialized program and load



Using ISeq.load_iseq



When should we compile?

- Compile timing
  - Use compiler explicitly
    - C/Java/… compilers
    - Gem install timing is good idea to kick it
  - Loading time (if not available, compile automatically)
    - Python (.pyc), Rubinius (.rbc)
    - Matz doesn’t like it

Where to store?

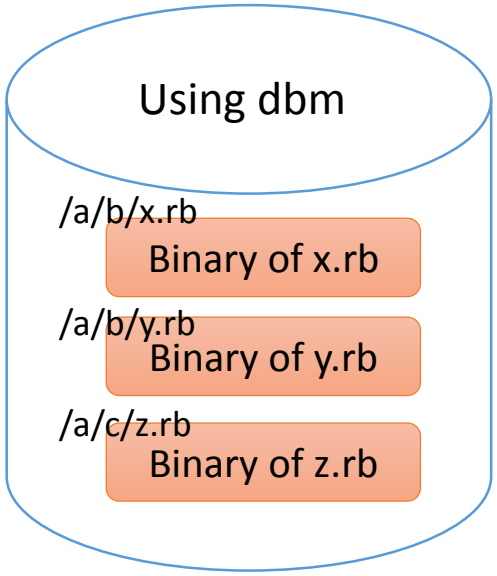
- “sample/iseq_load.rb” provides 3 type of repository

Store compiled binary in the same directory

```
/a/b/x.rb, x.rb.yarb
     y.rb, y.rb.yarb
   c/z.rb, z.rb.yarb
```

Store compiled binary in the specified directory

```
/a/b/x.rb, y.rb
   c/z.rb
/repos/a_b_x.rb.yarb
       a_b_y.rb.yarb
       a_c_z.rb.yarb
```



Usage of iseq_load.rb

- $ ruby iseq_load.rb [file or dir]
  - Compile, serialize and store specified file or files in directories (dir/**/*.rb)
- $ ruby -r iseq_load [script]
  - Enable loader
  - Load stored files if possible
- Setting by environment variables
- See iseq_load.rb for details

NOTE: Experimental feature

- All of features are introduced as “Experimental”
  - We don’t guarantee to keep these interface (methods)
  - We don’t guarantee to keep binary format
- No verifier so that loading modified/broken binary causes critical problem

→ Do not load any binary data provided by others

Enjoy hacking your great Ruby program cache!

Evaluation

Evaluation

- Measure loading time of same script 1,000 times
  - Use remove_const to cleanup each loading
  - Choose from lib/*.rb

Target script	Lines	Size (KB)
resolv.rb	2,855	73
csv.rb	2,346	83
fileutils.rb	1,761	48
forwardable.rb	290	8

Evaluation
Loading time (x1,000)

	Normal (sec)	Load (sec)	Lazy load (sec)
resolve.rb	13.19	3.92 (x3.36)	2.42 (x5.45)
csv.rb	7.88	4.19 (x1.88)	2.85 (x2.76)
fileutils.rb	8.55	4.64 (x1.84)	3.61 (x2.37)
forwardable.rb	0.48	0.18 (x2.67)	0.12 (x4.00)

☺ 5 times faster on resolv.rb seems good

☹ Nobody load resolv.rb 1,000 times

Evaluation
Compiled binary size

Target script	Lines	Script size (KB)	Binary size (KB)
resolv.rb	2,855	73	337
csv.rb	2,346	83	170
fileutils.rb	1,761	48	202
forwardable.rb	290	8	14

Evaluation
Rails launch time

- Loading time of sample simple Rails application
  - $ rails r “”

	Normal (sec)	Load (sec)	Lazy load (sec)
w/o bundle	1.89	1.42 (x1.33)	1.37 (x1.38)
w/ bundle	2.23	1.85 (x1.21)	1.85 (x1.21)

Future work

- Reduce memory consumption by memory sharing with mmap (and so on)
- Reduce binary size with some techniques
  - Smart serialization technique
  - Compaction technique
- And more…

Does anyone have an interest?

They may be worth hack topics.

Summary

- Introduced new “script serializer and deserializer”
- You can try this feature with Ruby 2.3 preview 2

Thank you for your attention

Koichi Sasada

<ko1@heroku.com>

salesforce heroku