Toward “more” efficient Ruby 2.1

Koichi Sasada

<ko1@heroku.com>

Heroku, Inc.

Agenda

- Ruby's rough history
- Ruby 2.1 new "internal" features
  - Internal object management hooks
    - Object allocation tracing
    - GC hooks
  - **RGenGC: Restricted Generational Garbage Collection ← Today's main topic**
- Ruby 2.1 expected "internal" features
  - Parallel sweeping
  - Sophisticated inline cache invalidation mechanism
  - Memory efficient string management

About this presentation

- This presentation is advanced version of my last presentation at RubyKaigi 2013 (May)
  - Talked in Japanese (with English slides)
  - Recycle presentation (≒ Good lazy programmer)

Slide PDF is http://rvm.jp/t.pdf

(temporary URL)

- I'm poor at English speaking
  - All contents I want to say are written in my slides
  - Please give me a question with **slow/clear/easy** English ☺

This presentation is NOT about

- Not about Rails application development
- Not about Programming language design
- Not about Mathematics
- Not abou Functional programming languages
- Not about Ruby programming language

Mainly about C programming language because it is about “C”Ruby

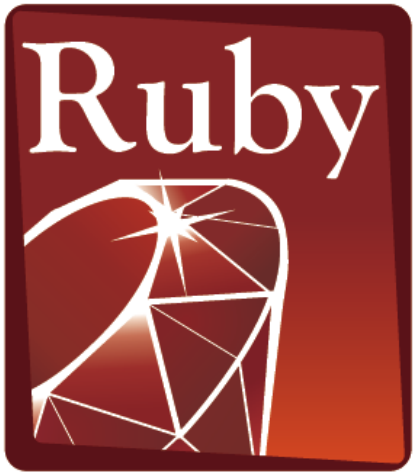
Who am I ?

- Koichi Sasada a.k.a ko1
- 笹田耕一 in Kanji character
- Japanese lesson: “1”
  - One in English
  - Mono in Greece
  - Eins in German
  - Un in French
  - Uno in Italian, Spanish
  - “Ichi” (“一” in Kanji) in Japanese
- I’m the first son of my parents

Who am I ?

heroku

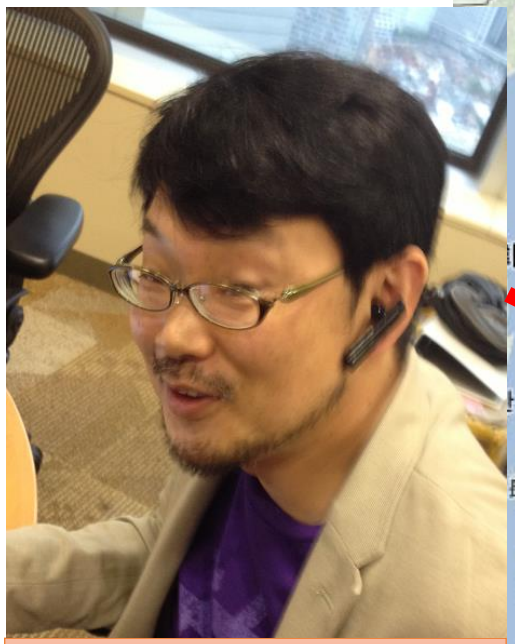
- Koichi Sasada
  - Matz team at Heroku, Inc.
    - Full-time CRuby developer
    - Working in Japan
  - CRuby/MRI committer
    - Virtual machine (YARV) from Ruby 1.9
    - YARV development since 2004/1/1



PROGRAMMING
Language

Matz team in Heroku

Nobu @ Tochigi
Patch monster



Matz @ Shimane
Title collector

ko1 @ Tokyo
EDD developer

Matz team at Heroku Hierarchy

Matz @ Shimane
Title collector

Communication
with Skype

ko1 @ Tokyo
EDD developer

Nobu @ Tochigi
Patch monster

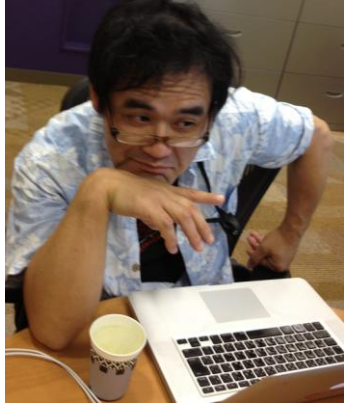
Matz
Title collector

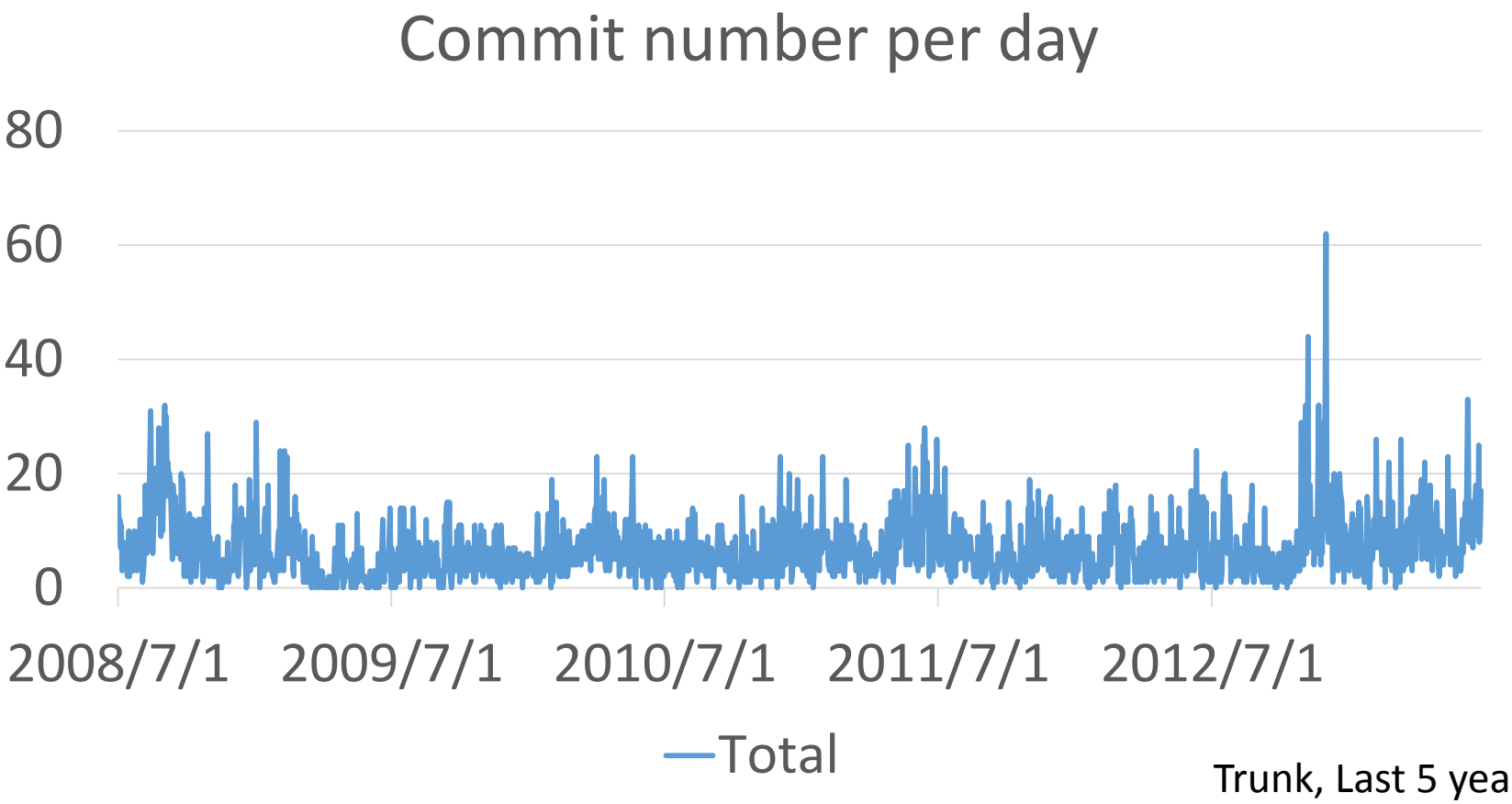
- He has so many (job) title
  - Chairman - Ruby Association
  - Fellow - NaCl
  - Chief architect, Ruby - Heroku
  - Research institute fellow – Rakuten
  - Chairman – NPO mruby Forum
  - Senior researcher – Kadokawa Ascii Research Lab
  - Visiting professor – Shimane University
  - Honorable citizen (living) – Matsue city
  - Honorable member – Nihon Ruby no Kai
  - …
- This margin is too narrow to contain

Nobu
Patch monster

- Great patch creator

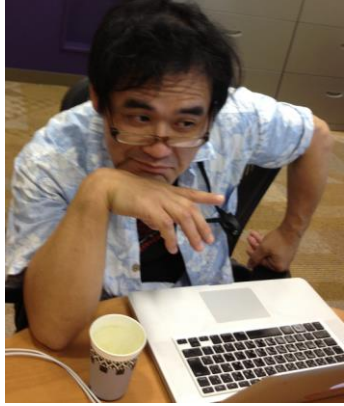
Nobu
Patch monster

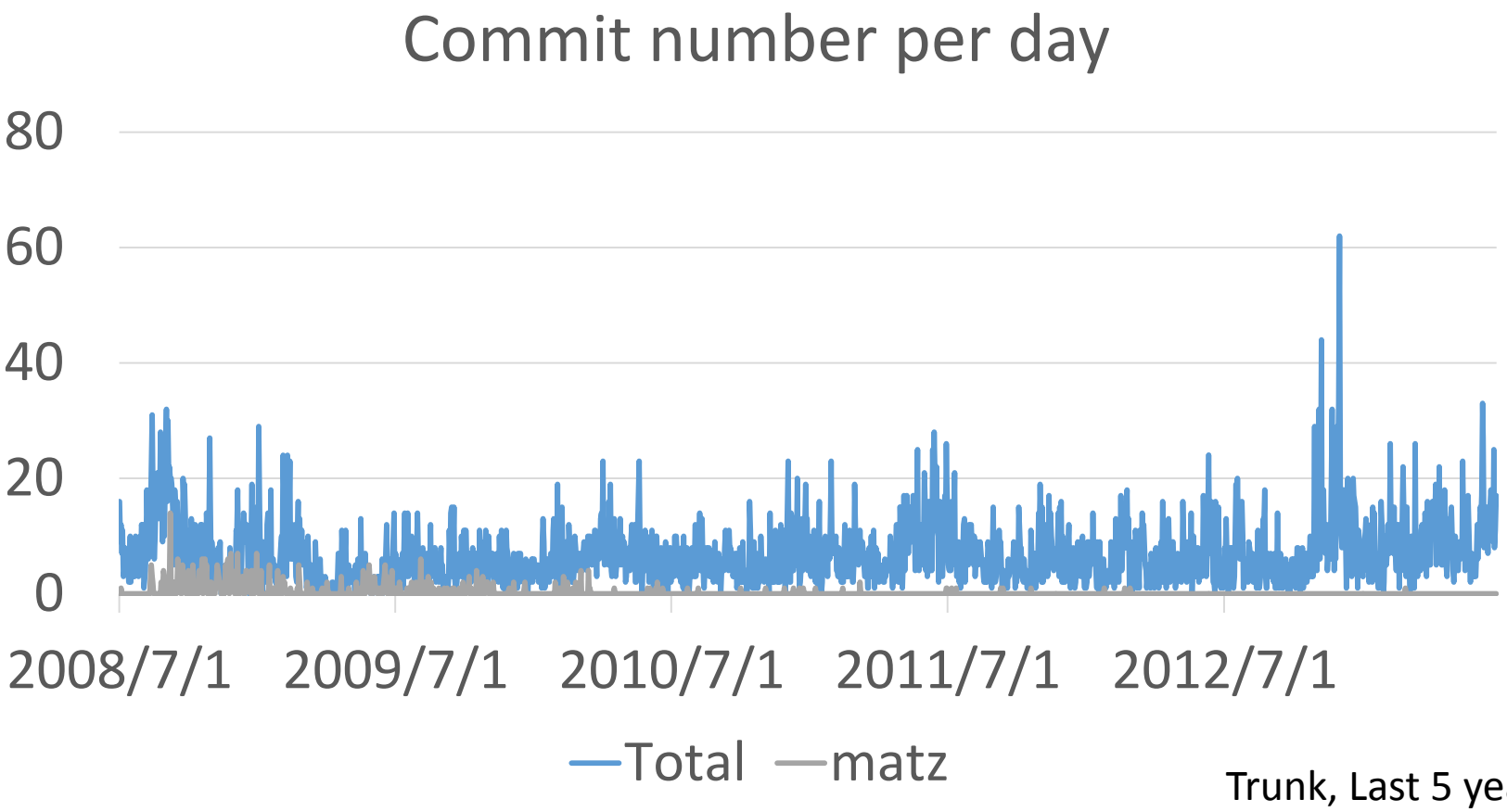




Trunk, Last 5 year

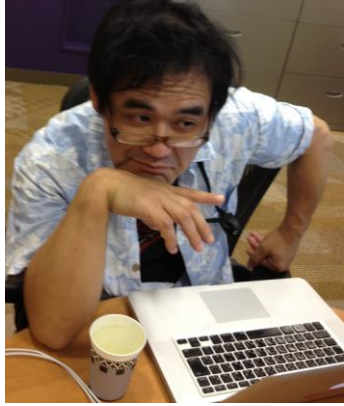
Nobu
Patch monster

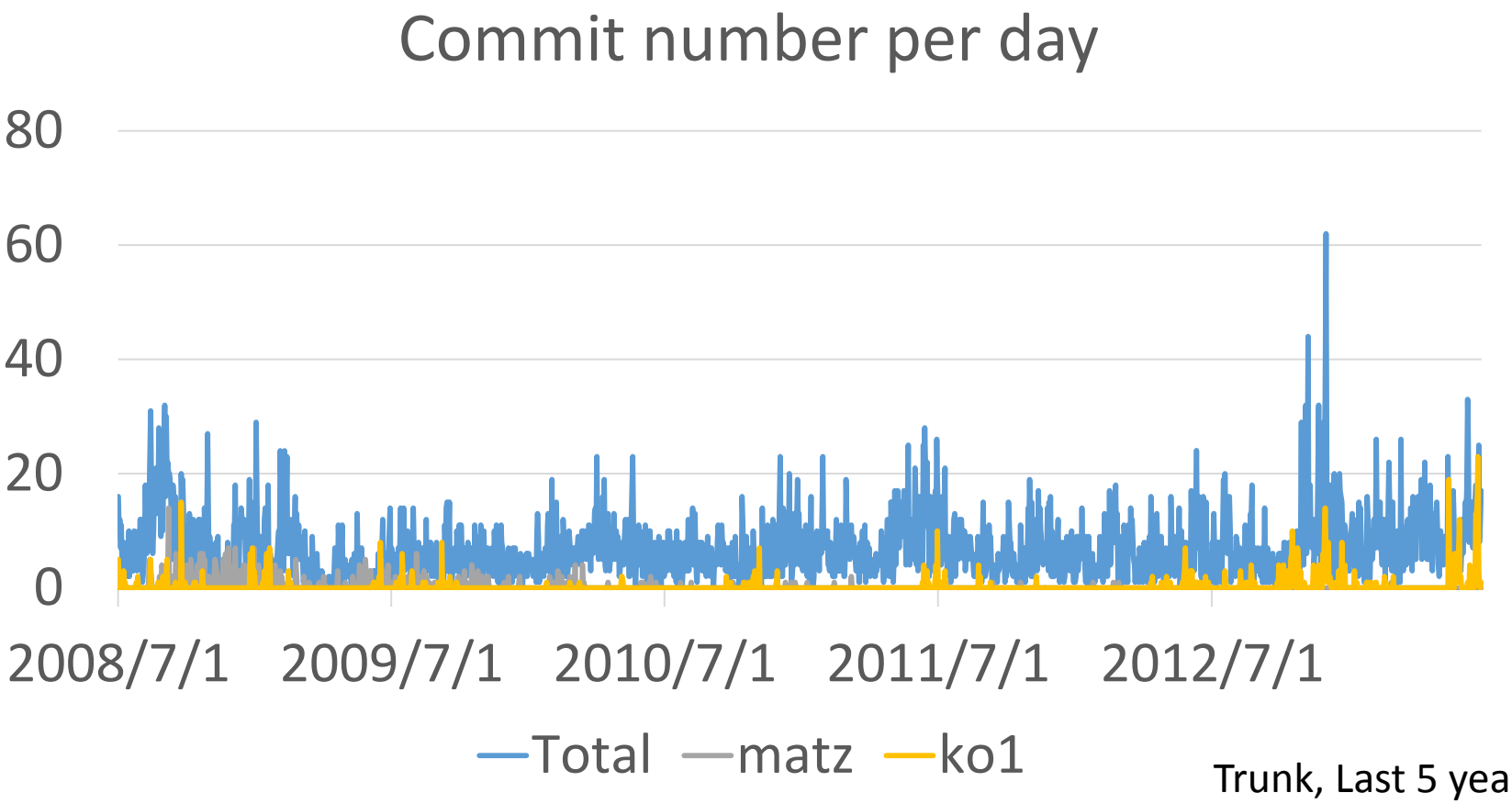




Trunk, Last 5 year

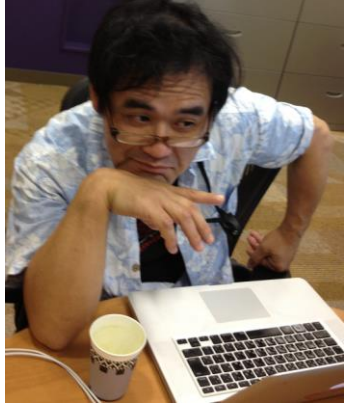
Nobu
Patch monster

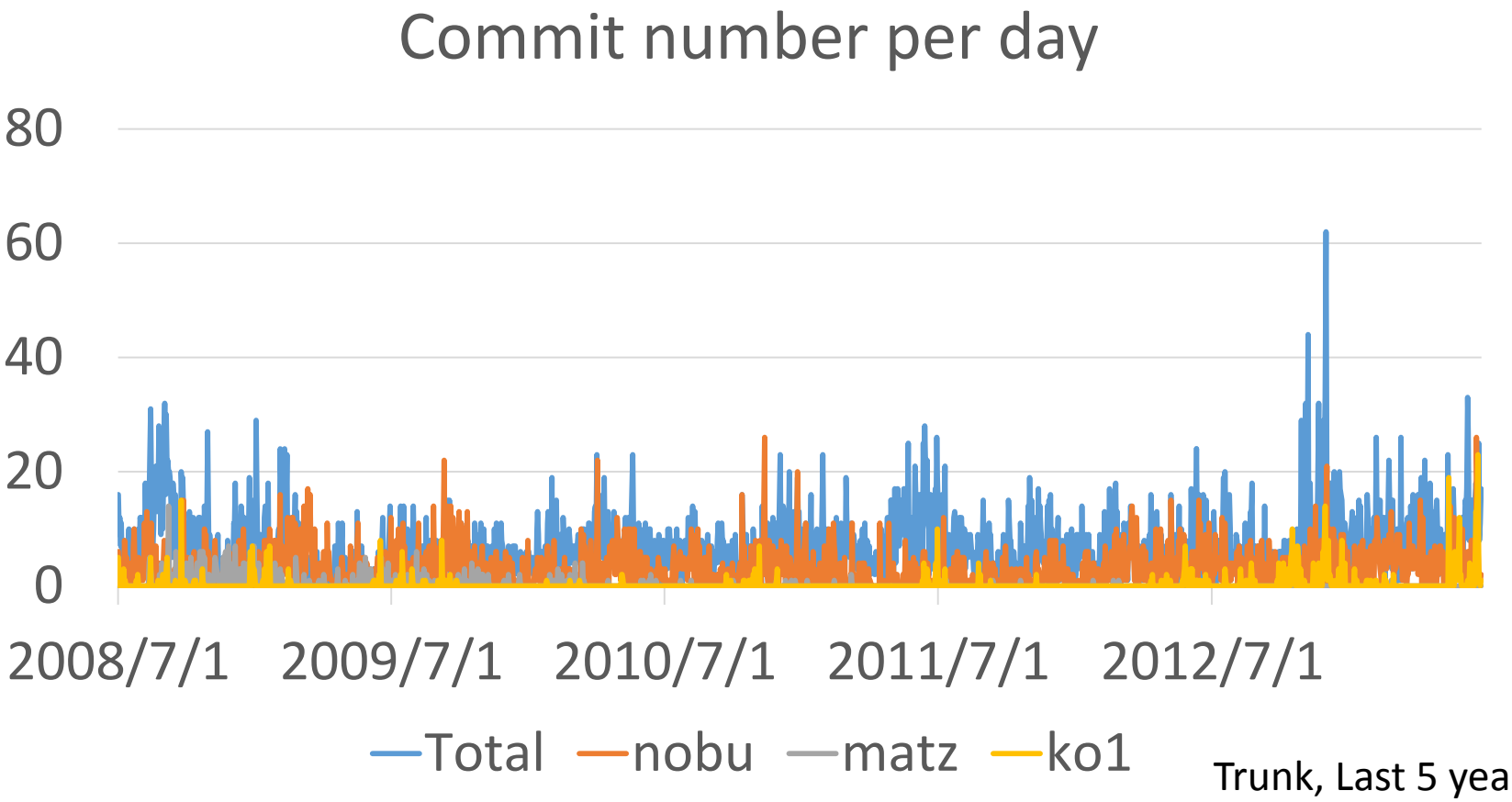




Trunk, Last 5 year

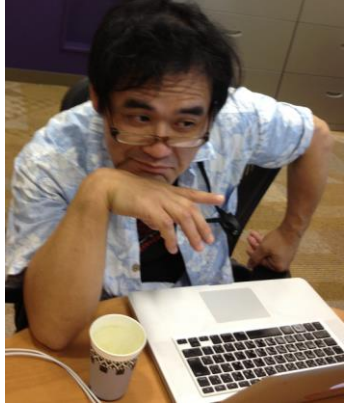
Nobu
Patch monster

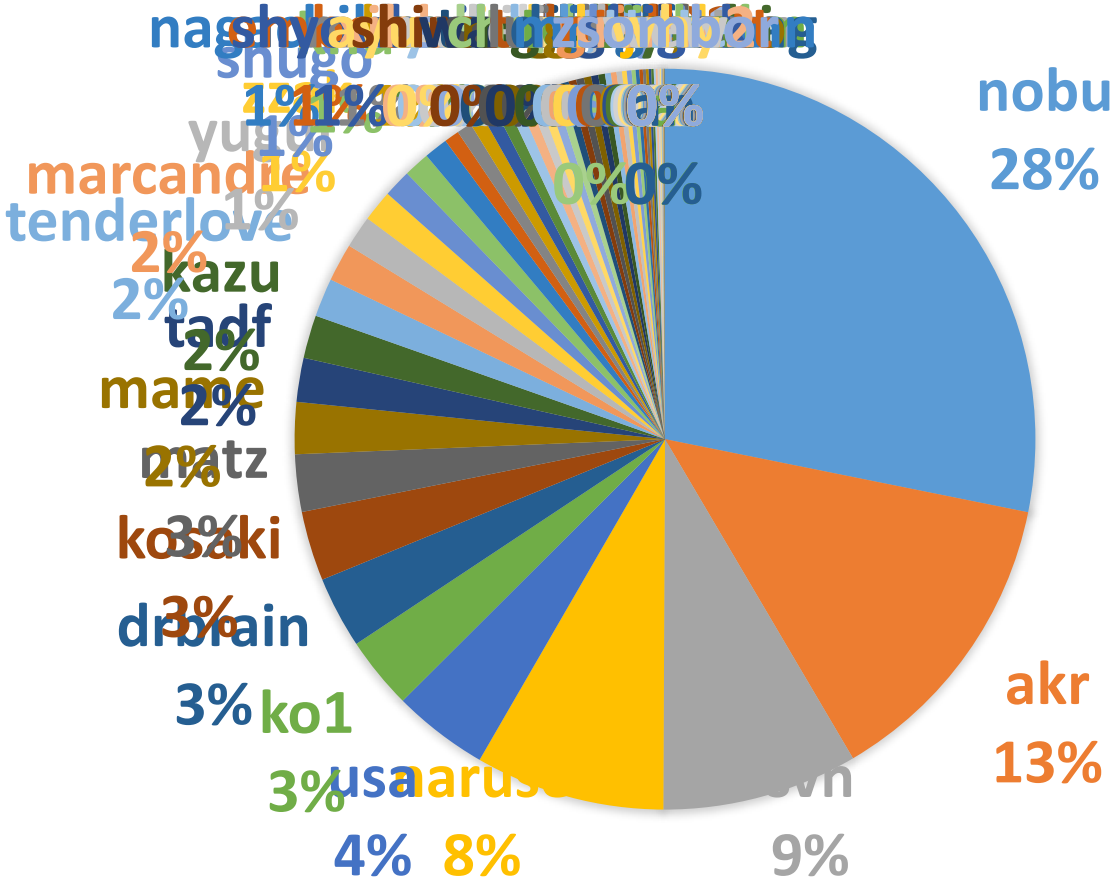




Trunk, Last 5 year

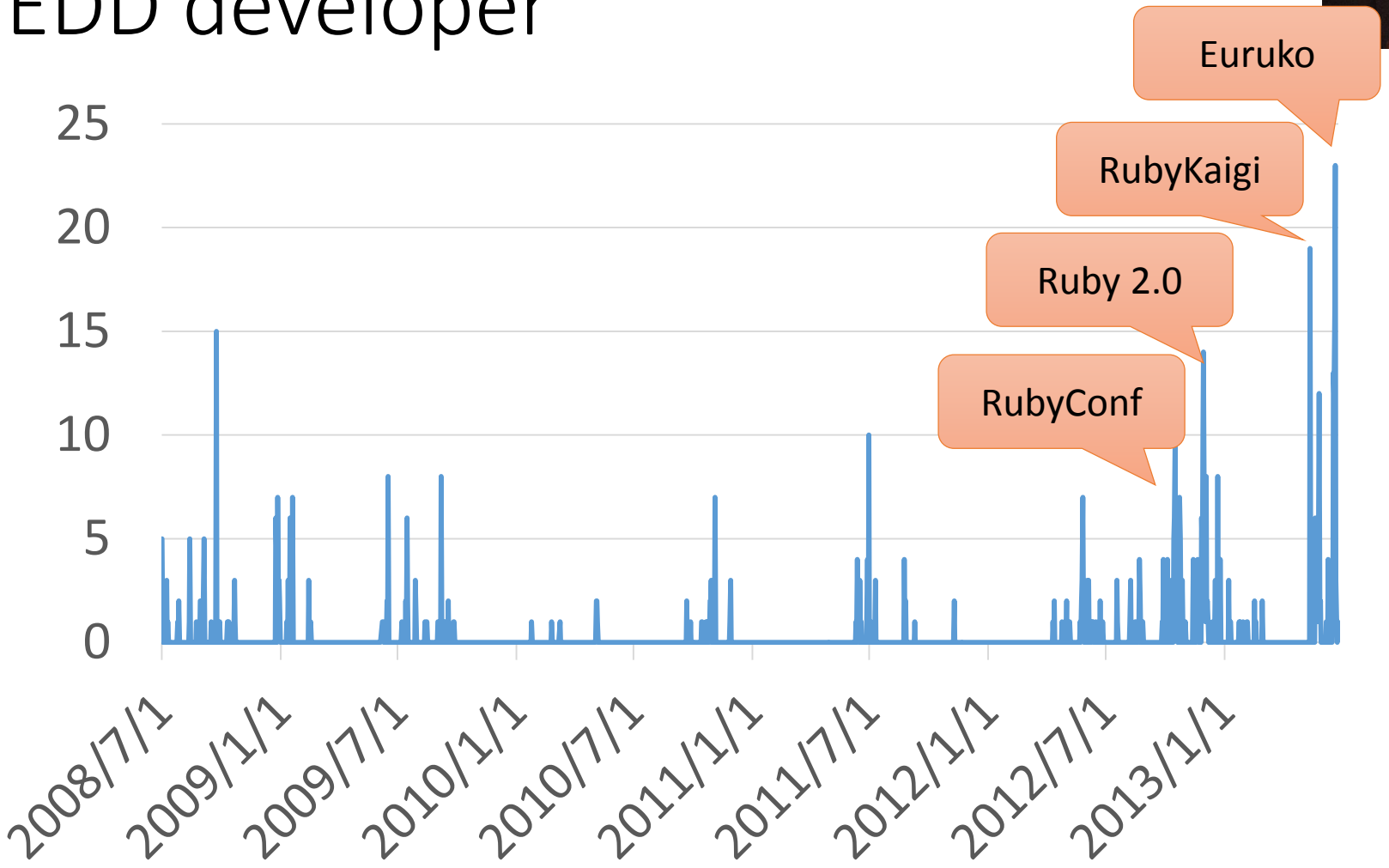
Nobu
Patch monster





Trunk, Last 5 year

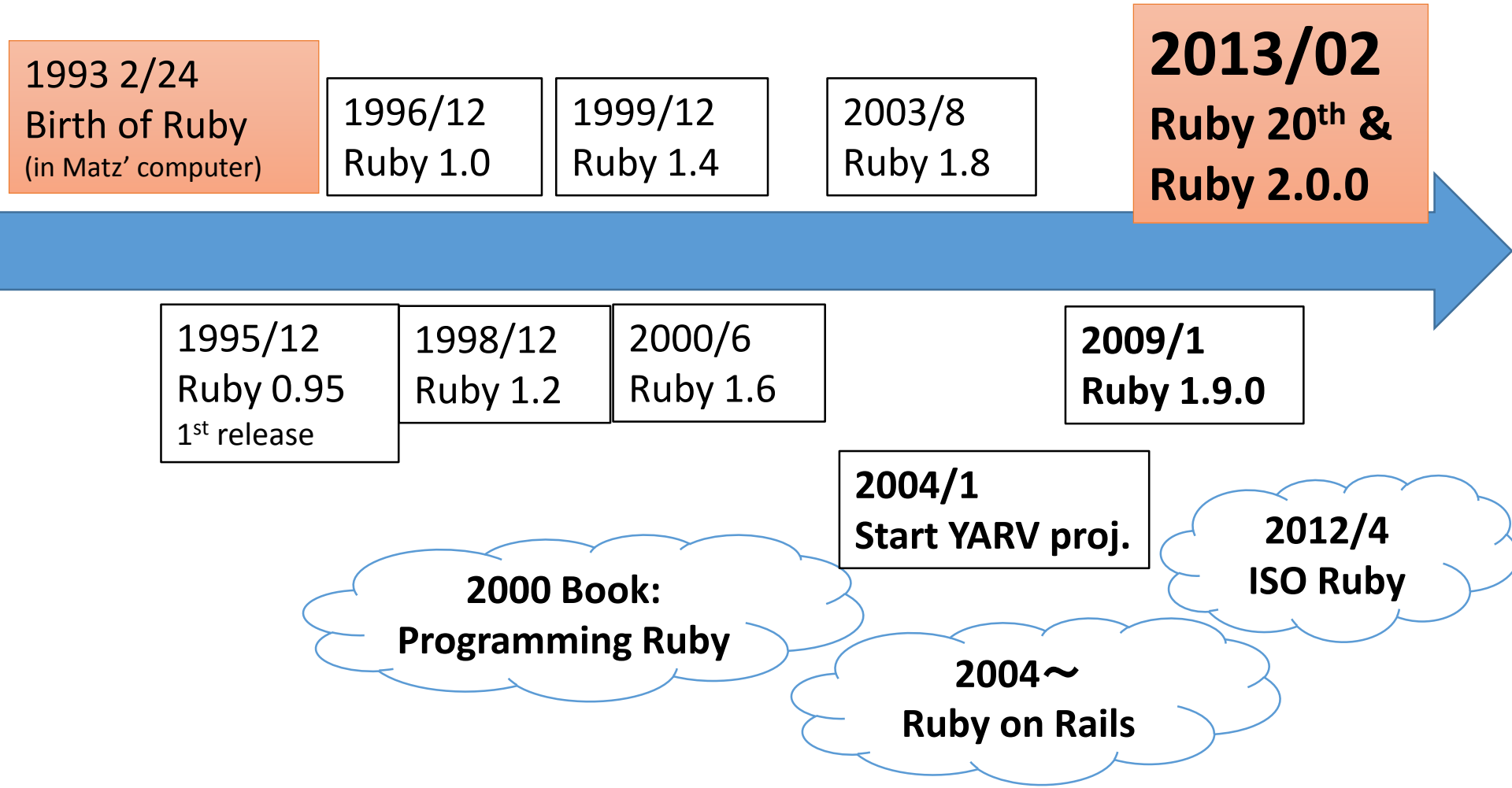
Ko1
EDD developer



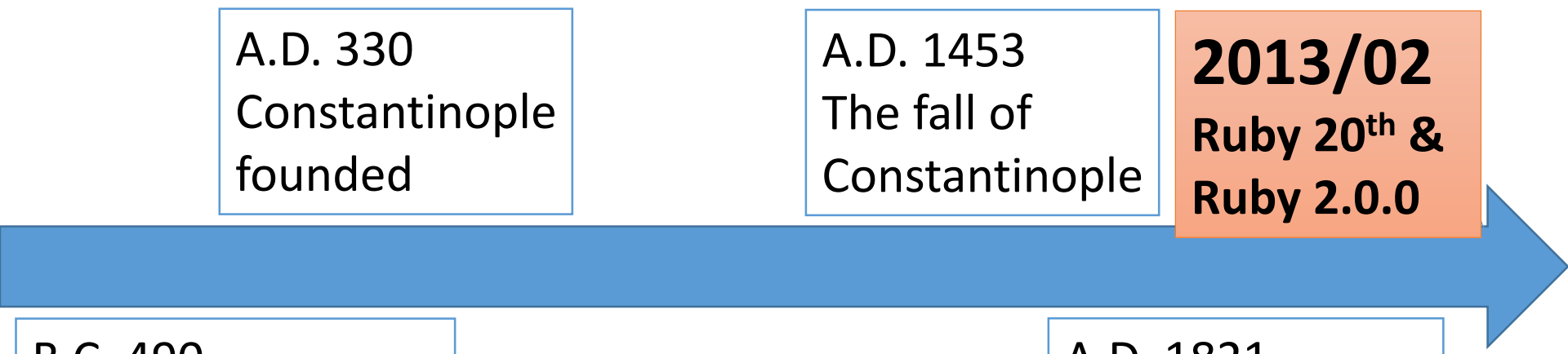
EDD: Event Driven Development

Brief history of Ruby

Brief history of Ruby



Brief history of Ruby



"20 years" is not so long!

(compare with Greece history)

ISO Ruby Standard

- Published at 2012/04
  - ISO/IEC 30170:2012 Information technology -- Programming languages – Ruby
  - http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=060&ics3=&csnumber=59579

  "ISO/IEC 30170:2012 specifies the syntax and semantics of the computer programming language Ruby, and the requirements for conforming Ruby processors, strictly conforming Ruby programs, and conforming Ruby programs."
- Hybrid 1.8 and 1.9
  - Difference parts are "undefined"

Ruby 2.0
Stable version

Ruby 2.0

- New features
  - Keyword arugments
  - Refinements
  - Module#prepend
- Ruby 2.0.0-p~~195~~p247 was already released

NEWS file of Ruby 2.0

Many new features!!

-*- rdoc -*-

= NEWS for Ruby 2.0.0

This document is a list of user visible feature changes made between

releases except for bug fixes.

Note that each entry is kept so brief that no reason behind or

reference information is supplied with. For a full list of changes

with all sufficient information, see the ChangeLog file.

== Changes since the 1.9.3 release

=== C API updates

* NUM2SHORT() and NUM2USHORT() added. They are similar to NUM2INT, but short.

* rb_newobj_of() and NEWOBJ_OF() added. They create a new object of a given class.

=== Library updates (outstanding ones only)

* builtin classes

  * Array

    * added method:

      * added Array#bsearch for binary search.

    * incompatible changes:

      * random parameter of Array#shuffle! and Array#sample now

        will be called with one argument, maximum value.

      * when given Range arguments, Array#values_at now returns nil for each

        value that is out-of-range.

  * Enumerable

    * added method:

      * added Enumerable#lazy method for lazy enumeration.

  * Enumerator

    * added method:

      * added Enumerator#size for lazy size evaluation.

    * extended method:

      * Enumerator.new accept an argument for lazy size evaluation.

  * ENV

    * aliased method:

      * ENV.to_h is a new alias for ENV.to_hash

  * Fiber

    * incompatible changes:

      * Fiber#resume cannot resume a fiber which invokes "Fiber#transfer".

  * File

    * extended method:

      * File.fnmatch? now expands braces in the pattern if

        File::FNM_EXTGLOB option is given.

  * GC

    * improvements:

      * introduced the bitmap marking which suppresses to copy a memory page

        with Copy-on-Write.

      * introduced the non-recursive marking which avoids unexpected stack overflow.

  * GC::Profiler

    * added method:

      * added GC::Profiler.raw_data which returns raw profile data for GC.

  * Hash

    * added method:

      * added Hash#to_h as explicit conversion method, like Array#to_a.

    * extended method:

      * Hash#default_proc= can be passed nil to clear the default proc.

  * Kernel

    * added method:

      * added Kernel#Hash conversion method like Array() or Float().

      * added Kernel#using, which imports refinements into the current scope.

        [experimental]

      * added Kernel#__dir__ which returns a current dirname.

      * added Kernel#caller_locations which returns an array of

        frame information objects.

    * extended method:

      * Kernel#warn accepts multiple args in like puts.

      * Kernel#caller accepts second optional argument 'n' which specify

        required caller size.

      * Kernel#to_enum and enum_for accept a block for lazy size evaluation.

    * incompatible changes:

      * system() and exec() closes non-standard file descriptors

        (The default of :close_others option is changed to true by default.)

      * respond_to? against a protected method now returns false unless

        the second argument is true.

      * __callee__ has returned to the original behavior, and now

        returns the called name but not the original name in an

        aliased method.

      * Kernel#inspect does not call #to_s anymore

        (it used to call redefined #to_s).

  * LoadError

    * added method:

      * added LoadError#path method to return the file name that could not be

        loaded.

  * Module

    * added method:

      * added Module#prepend which is similar to Module#include,

        however a method in the prepended module overrides the

        corresponding method in the prepending module.

      * added Module#refine, which extends a class or module locally.

        [experimental]

      * added Module#refinements, which returns refinements defined in the

        receiver. [experimental]

      * added Module#using, which imports refinements into the receiver.

        [experimental]

      * extended method:

        * Module#define_method accepts a UnboundMethod from a Module.

        * Module#const_get accepts a qualified constant string, e.g.

          Object.const_get("Foo::Bar::Baz")

      * Mutex

        * added method:

          * added Mutex#owned? which returns the mutex is held by current

            thread or not. [experimental]

        * incompatible changes:

          * Mutex#lock, Mutex#unlock, Mutex#try_lock, Mutex#synchronize

            and Mutex#sleep are no longer allowed to be used from trap handler

            and raise a ThreadError in such case.

      * Mutex#sleep may spurious wakeup. Check after wakeup.

  * NilClass

    * added method:

      * added nil.to_h which returns {}

  * Process

    * added method:

      * added getsid for getting session id (unix only).

  * Range

    * added method:

      * added Range#size for lazy size evaluation.

      * added Range#bsearch for binary search.

  * Signal

    * added method:

      * added Signal.signame which returns signal name

    * incompatible changes:

      * Signal.trap raises ArgumentError for

        when SEGV, BUS, ILL, FPE or VTALRM

        are specified.

  * String

    * added method:

      * added String#b returning a copied string whose encoding is ASCII-8BIT.

    * change return value:

      * String#lines now returns an array instead of an enumerator.

      * String#chars now returns an array instead of an enumerator.

      * String#codepoints now returns an array instead of an enumerator.

      * String#bytes now returns an array instead of an enumerator.

  * Struct

    * added method:

      * added Struct#to_h returning values with keys corresponding to the

        instance variable names.

  * Thread

    * added method:

      * added Thread#thread_variable_get for getting thread local variables

        (these are different than Fiber local variables).

      * added Thread#thread_variable_set for setting thread local variables.

      * added Thread#thread_variables for getting a list of the thread local

        variable keys.

      * added Thread#thread_variable? for testing to see if a particular thread

        variable has been set.

      * added Thread#backtrace_locations which returns similar information of

        Kernel#caller_locations.

    * incompatible changes:

      * Thread#join and Thread#value now raises a ThreadError if target thread

        is the current or main thread.

  * Time

    * change return value:

      * Time#to_s returned encoding defaults to US-ASCII but automatically

        changes to Encoding.default_internal if it is

        not nil.

  * TracePoint

    * new class. This class is replacement of set_trace_func.

      Easy to use and efficient implementation.

  * toplevel

    * added method:

      * added main.define_method which defines a global function.

  * cgi

    * Add HTML5 tag maker.

    * CGI#header has been renamed to CGI#http_header and

      aliased to CGI#header.

    * When HTML5 tagmaker called, overwrite CGI#header,

      CGI#header function is to create a <header> element.

  * iconv

    * iconv has been removed. Use String#encode instead.

  * io/wait

    * new features:

      * added IO#wait_writable method.

      * added IO#wait_readable method as alias of IO#wait.

  * net/http

    * new features:

      * Proxies are now automatically detected from the http_proxy environment

        variable. See Net::HTTP::new for details.

      * gzip and deflate compression are now requested for all requests by

        default. See Net::HTTP for details.

      * SSL sessions are now reused across connections for a single instance.

        This speeds up connection by using a previously negotiated session.

    * new methods:

      * Net::HTTP#local_host

      * Net::HTTP#local_host=

      * Net::HTTP#local_port

      * Net::HTTP#local_port=

    * extended method:

      * Net::HTTP#connect uses local_host and local_port if specified.

  * net/imap

    * new methods:

      * Net::IMAP.default_port

      * Net::IMAP.default_imap_port

      * Net::IMAP.default_tls_port

      * Net::IMAP.default_ssl_port

      * Net::IMAP.default_imaps_port

  * objspace

    * new method:

      * ObjectSpace.reachable_objects_from(obj)

  * openssl

    * Consistently raise an error when trying to encode nil values. All instances

      of OpenSSL::ASN1::Primitive now raise TypeError when calling to_der on an

      instance whose value is nil. All instances of OpenSSL::ASN1::Constructive

      raise NoMethodError in the same case. Constructing such values is still

      permitted.

    * TLS 1.1 & 1.2 support by setting OpenSSL::SSL::SSLContext#ssl_version to

      :TLSv1_2, :TLSv1_2_server, :TLSv1_2_client or :TLSv1_1, :TLSv1_1_server

      :TLSv1_1_client. The version being effectively used can be queried

      with OpenSSL::SSL#ssl_version. Furthermore, it is also possible to

      blacklist the new TLS versions with OpenSSL::SSL::OP_NO_TLSv1_1 and

      OpenSSL::SSL::OP_NO_TLSv1_2.

    * Added OpenSSL::SSL::SSLContext#renegotiation_cb. A user-defined callback

      may be set which gets called whenever a new handshake is negotiated. This

      also allows to programmatically decline (client) renegotiation attempts.

    * Support for "0/n" splitting of records as BEAST mitigation via

      OpenSSL::SSL::OP_DONT_INSERT_EMPTY_FRAGMENTS.

    * OpenSSL requires passwords for decrypting PEM-encoded files to be at least

      four characters long. This led to awkward situations where an export with

      a password with fewer than four characters was possible, but accessing the

      file afterwards failed. OpenSSL::PKey::RSA, OpenSSL::PKey::DSA and

      OpenSSL::PKey::EC therefore now enforce the same check when exporting a

      private key to PEM with a password - it has to be at least four characters

      long.

    * SSL/TLS support for the Next Protocol Negotiation extension. Supported

      with OpenSSL 1.0.1 and higher.

    * OpenSSL::OPENSSL_FIPS allows client applications to detect whether OpenSSL

      is running in FIPS mode and to react to the

      fatal mode switch (OpenSSL

      might [...]

  * ostruct

    * new methods:

      * OpenStruct#each_pair

      * OpenStruct#eql?

      * OpenStruct#hash

      * OpenStruct#to_h converts the struct to a hash.

    * extended method:

      * OpenStruct.new also accepts an OpenStruct / Struct.

  * pathname

    * extended method:

      * Pathname#find returns an enumerator if no block is given.

  * rake

    * rake has been updated to version 0.9.5.

      This version is backwards-compatible with previous rake versions and

      contains many bug fixes.

      See

      http://rake.rubyforge.org/doc/release_notes/rake-0_9_5_rdoc.html for a list

      of changes in rake 0.9.3, 0.9.4 and 0.9.5.

  * rdoc

    * rdoc has been updated to version 4.0

      This version is largely backwards-compatible with previous rdoc versions.

      The most notable change is an update to the ri data format (ri data must

      be regenerated for gems shared across rdoc versions). Further API changes

      are internal and won't affect most users.

      See https://github.com/rdoc/rdoc/blob/master/History.rdoc for a list of

      changes in rdoc 4.0.

  * resolv

    * new methods:

      * Resolv::DNS#timeouts=

      * Resolv::DNS::Config#timeouts=

  * rexml

    * REXML::Document#write supports Hash arguments.

      * REXML::Document#write supports new :encoding option. It changes

        XML document encoding. Without :encoding option, encoding in

        XML declaration is used for XML document encoding.

  * RubyGems

    * Updated to 2.0.0.preview2

      RubyGems 2.0.0 features the following improvements:

      * Improved support for default gems shipping with ruby 2.0.0+

      * A gem can have arbitrary metadata through Gem::Specification#metadata

      * `gem search` now defaults to --remote and is anchored like gem list.

      * Added --document to replace --rdoc and --ri. Use --no-document to

        disable documentation, --document=rdoc to only generate rdoc.

      * Only ri-format documentation is generated by default.

      * `gem server` uses RDoc::Servlet from RDoc 4.0 to generate HTML

        documentation.

      For an expanded list of updates and bug fixes see:

      https://github.com/rubygems/rubygems/blob/master/History.txt

  * shellwords

    * Shellwords#shellescape() now stringifies the given object using to_s.

    * Shellwords#shelljoin() accepts non-string objects in the given

      array, each of which is stringified using to_s.

  * syslog

    * Added Syslog::Logger which provides a Logger API atop Syslog.

    * Syslog::Priority, Syslog::Level, Syslog::Option and Syslog::Macros

      are introduced for easy detection of available constants on a

      running system.

  * tmpdir

    * incompatible changes:

      * Dir.mktmpdir uses FileUtils.remove_entry instead of

        FileUtils.remove_entry_secure. This means that applications should not

        change the permission of the created temporary directory to make

        accessible from other users.

  * yaml

    * Syck has been removed. YAML now completely depends on libyaml being

      installed.

  * zlib

    * Added streaming support for Zlib::Inflate and Zlib::Deflate. This allows

      processing of a stream without the use of large amounts of memory.

    * Added support for the new deflate strategies Zlib::RLE and Zlib::FIXED.

    * Zlib streams are now processed without the GVL. This allows gzip, zlib and

      deflate streams to be processed in parallel.

=== Language changes

* Added %i and %I for symbol list creation (similar to %w and %W).

* Default source encoding is changed to UTF-8. (was US-ASCII)

=== Compatibility issues (excluding feature bug fixes)

* Array#values_at

  See above.

* String#lines

* String#chars

* String#codepoints

* String#bytes

  These methods no longer return an Enumerator, although passing a

  block is still supported for backwards compatibility.

  Code like str.lines.with_index(1) { |line, lineno| ... } no longer

  works because str.lines returns an array. Replace lines with

  each_line in such cases.

* Signal.trap

  See above.

* Merge Onigmo.

  https://github.com/k-takata/Onigmo

* The :close_others option is true by default for system() and exec().

  Also, the close-on-exec flag is set by default for all new file descriptors.

  This means file descriptors doesn't inherit to spawned process unless

  explicitly requested such as system(..., fd=>fd).

* Kernel#respond_to? against a protected method now returns false

  unless the second argument is true.

* Dir.mktmpdir in lib/tmpdir.rb

  See above.

* OpenStruct new methods can conflict with custom attributes named

  "each_pair", "eql?", "hash" or "to_h".

* Thread#join, Thread#value

  See above.

* Mutex#lock, Mutex#unlock, Mutex#try_lock, Mutex#synchronize and Mutex#sleep

  See above.

「Rubyは言語として2.0でほぼ完成」、まつもとゆきひろ氏が講演

2013/02/14
安東 一真＝日経Linux

記事一覧へ >>　いいね！ 108　ツイート 257

「Rubyはバージョン2.0で、言語としてほぼ完成した」――。東京・目黒雅叙園で2月15日まで開催している「Developers Summit 2013」で、Rubyの生みの親であるまつもとゆきひろ氏（**写真**）はこう宣言した。

Ruby 2.0は、Ruby生誕20周年を記念して、2013年2月24日にリリースする予定の新バージョン。まつもと氏は講演の中で、バージョン2.0の新機能を披露するとともに、

写真●まつもとゆきひろ氏
[画像のクリックで拡大表示]

Matz said “Ruby is almost matured as a programming language with 2.0 ”

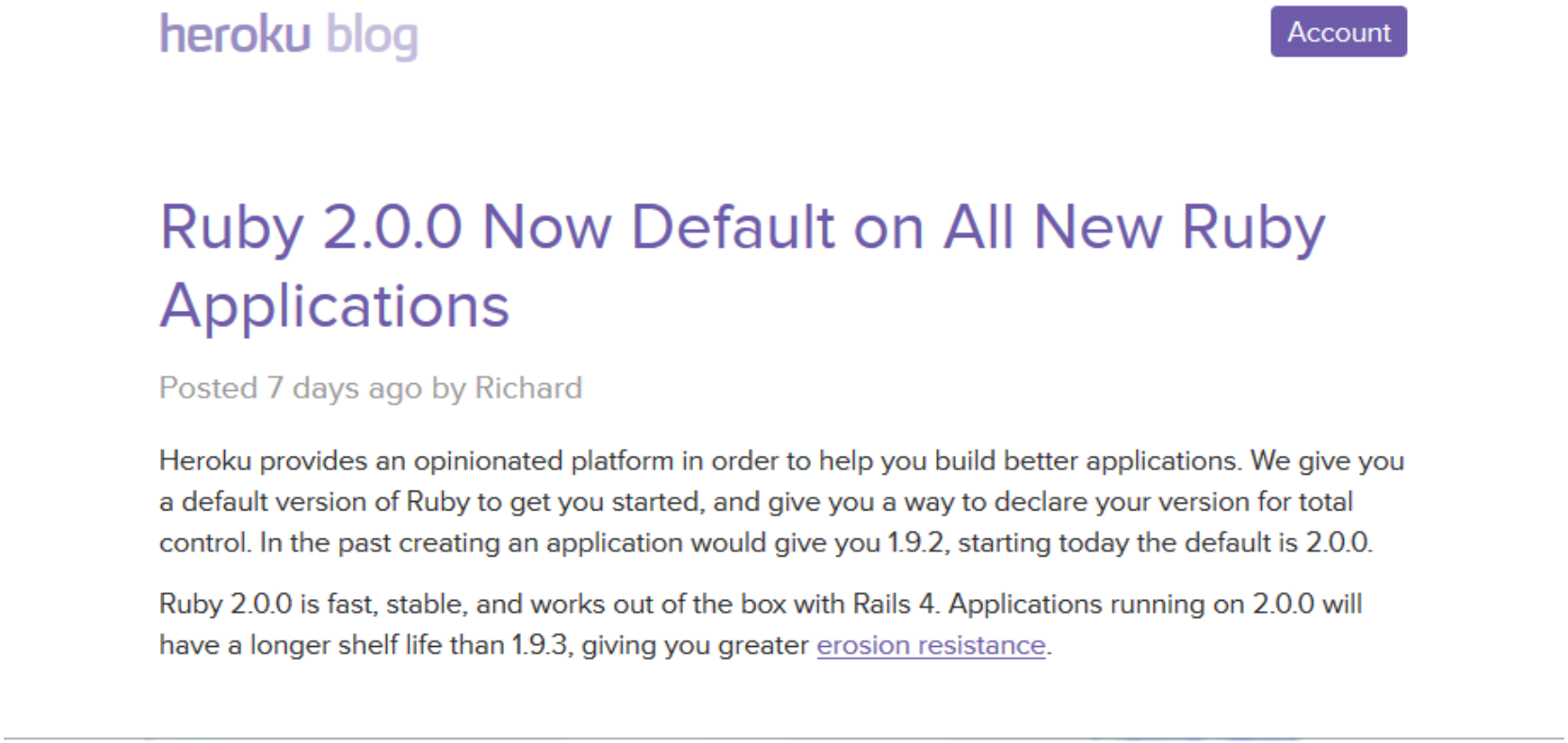
http://itpro.nikkeibp.co.jp/article/NEWS/20130214/456322/

Ruby versions

- Which version of Ruby (MRI) do you use?
  1. Ruby 1.8.7
  2. Ruby 1.9.2
  3. Ruby 1.9.3
  4. Ruby 2.0.0 p0
  5. Ruby 2.0.0 p195
  6. Ruby 2.0.0 p247

PR

Ruby 2.0.0 is default at Heroku



heroku blog

Account

Ruby 2.0.0 Now Default on All New Ruby Applications

Posted 7 days ago by Richard

Heroku provides an opinionated platform in order to help you build better applications. We give you a default version of Ruby to get you started, and give you a way to declare your version for total control. In the past creating an application would give you 1.9.2, starting today the default is 2.0.0.

Ruby 2.0.0 is fast, stable, and works out of the box with Rails 4. Applications running on 2.0.0 will have a longer shelf life than 1.9.3, giving you greater erosion resistance.

https://blog.heroku.com/archives/2013/6/17/ruby-2-default-new-aps

Rubyist Magazine
Ruby 2.0 Special articles

Ruby 2.0.0 Release special articles

B! 1

- About Ruby 2.0.0 Release special articles
- Messages from Rubyists
  - Message from Matz
  - Ruby 2.0 on Rails
  - Change something silently
  - Message from Dave Thomas
    - Message
    - Favorite Feature
  - Message from Charles Oliver Nutter
  - Message from Thomas E Enebo

http://magazine.rubyist.net/?Ruby200SpecialEn

Ruby 2.1
Next version

Ruby 2.1 release plan announcement

"I, Naruse, take over the release manager of Ruby 2.1.0 from mame. ***Ruby 2.1.0 is planed to release in 2013-12-25.*** *I'm planning to call for feature proposals soon like 2.0.0 [ruby-core:45474], so if you have a suggestion you should begin preparing the proposal."*

- [ruby-core:54726] Announce take over the release manager of Ruby 2.1.0

by NARUSE, Yui

2013/12/25!

http://www.flickr.com/photos/htakashi/5285103341/ by Takashi Hososhima

Ruby 2.1 schedule

Events are important for
EDD (Event Driven Development) Developers

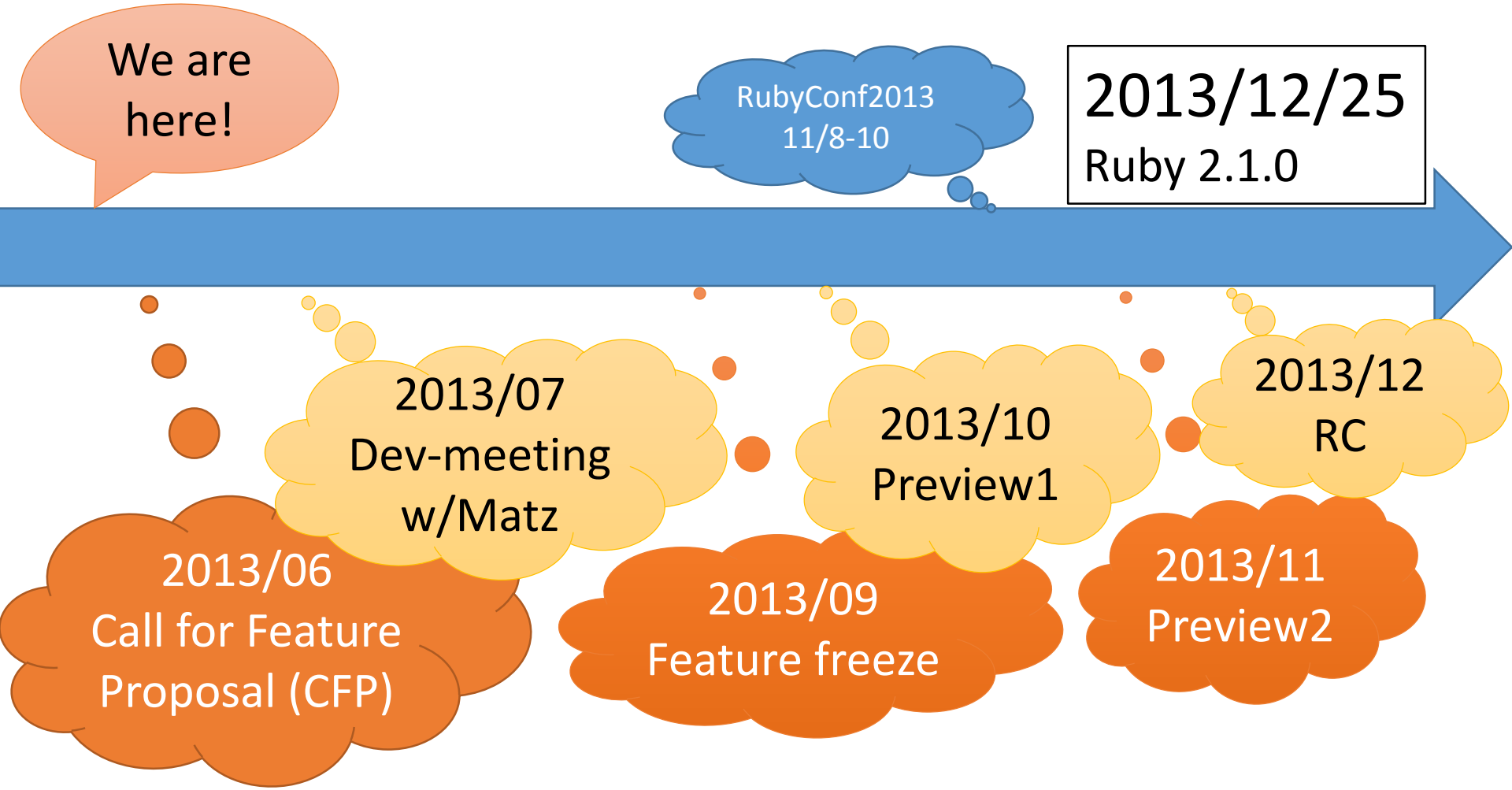
Ruby 2.1 release plan announcement

"I, Naruse, take over the release manager of Ruby 2.1.0 from mame. Ruby 2.1.0 is planed to release in 2013-12-25. ***I'm planning to call for feature proposals soon like 2.0.0 [ruby-core:45474], so if you have a suggestion you should begin preparing the proposal.****"*

- [ruby-core:54726] Announce take over the release manager of Ruby 2.1.0

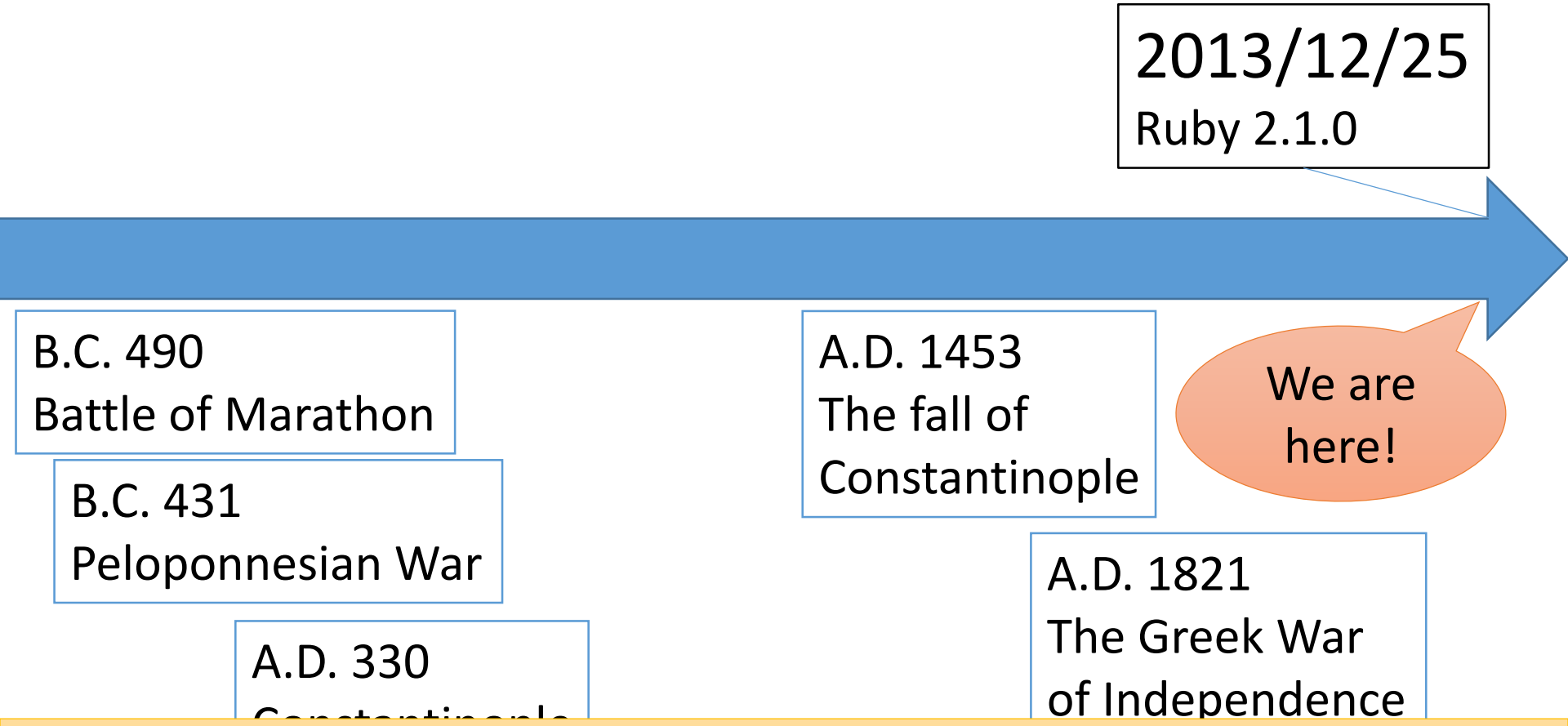
by NARUSE, Yui

Ruby 2.1 schedule (more)



https://bugs.ruby-lang.org/projects/ruby-trunk/wiki/ReleaseEngineering210

Ruby 2.1 schedule (more^2)



Ruby 2.1 will be release Immediately!

Ruby 2.1

- New features

```
# -*- rdoc -*-

= NEWS for Ruby 2.1.0

This document is a list of user visible feature changes made between
releases except for bug fixes.

Note that each entry is kept so brief that no reason behind or
reference information is supplied with.  For a full list of changes
with all sufficient information, see the ChangeLog file.

== Changes since the 2.0.0 release

=== Language changes
=== Core classes updates (outstanding ones only)

* GC
  * added environment variable:
    * RUBY_HEAP_SLOTS_GROWTH_FACTOR: growth rate of the heap.

* IO
  * extended methods:
    * IO#seek accepts symbols (:CUR, :END, :SET) for 2nd argument.

* Kernel
  * New methods:
    * Kernel#singleton_method

* Mutex
  * misc
    * Mutex#owned? is no longer experimental.

* String
  * New methods:
    * String#scrub and String#scrub! verify and fix invalid byte sequence.
  * extended methods:
    * If invalid: :replace is specified for String#encode, replace
      invalid byte sequence even if the destination encoding equals to
      the source encoding.

* pack/unpack (Array/String)
  * Q! and q! directives for long long type if platform has the type.

=== Core classes compatibility issues (excluding feature bug fixes)

* IO
  * incompatible changes:
    * open ignore internal encoding if external encoding is ASCII-8BIT.

* Module#ancestors

  The ancestors of a singleton class now include singleton classes,
  in particular itself.

=== Stdlib updates (outstanding ones only)

* Digest
  * extended methods:
    * Digest::Class.file takes optional arguments for its constructor

* Matrix
  * Added Vector#cross_product.

* Net::SMTP
  * Added Net::SMTP#rset to implement the RSET command

* Pathname
  * New methods:
    * Pathname#write
    * Pathname#binwrite

* OpenSSL::BN
  * extended methods:
    * OpenSSL::BN.new allows Fixnum/Bignum argument.

* open-uri
  * Support multiple fields with same field name (like Set-Cookie).

* Resolv
  * New methods:
    * Resolv::DNS.fetch_resource
  * One-shot multicast DNS support
  * Support LOC resources

* Rinda::RingServer, Rinda::RingFinger
  * Rinda now supports multicast sockets.  See Rinda::RingServer and
    Rinda::RingFinger for details.

* Socket
  * New methods:
    * Socket.getifaddrs

* StringScanner
  * extended methods:
    * StringScanner#[] supports named captures.

* Tempfile
  * New methods:
    * Tempfile.create

=== Stdlib compatibility issues (excluding feature bug fixes)

* URI
  * incompatible changes:
    * URI.decode_www_form follows current WHATWG URL Standard.
      It gets encoding argument to specify the character encoding.
      It now allows loose percent encoded strings, but denies ;-separator.
    * URI.encode_www_form follows current WHATWG URL Standard.
      It gets encoding argument to convert before percent encode.
      UTF-16 strings aren't converted to UTF-8 before percent encode by default.

=== C API updates
```

See NEWS file

Now, much smaller than Ruby 2.0

Ruby 2.1 features

- Refine m17n introduced from Ruby 1.9
  - String#scrub, String#scrub!
    - Verify and fix invalid byte sequence.
  - I heard Matz has some ideas.

- Refine features introduced from Ruby 2.0
  - Keyword arguments
  - Refinements
  - Module#prepend

Back to Ruby 2.0

Quote about Ruby 2.0 from Heroku blog

heroku

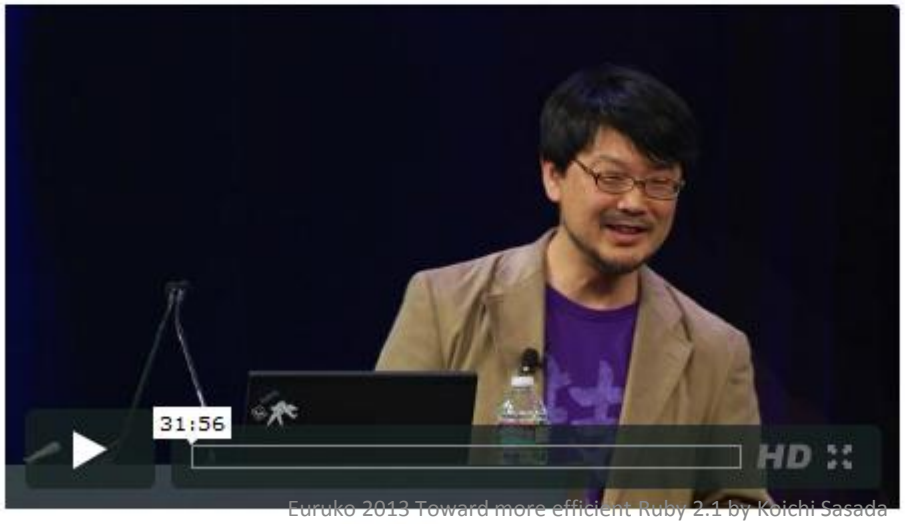
How it Works | Pricing | Add-ons | Dev Ce

Blog

Matz on Ruby 2.0 at Heroku's Waza

by Craig - Mar 06

Matz, the creator of Ruby, spoke at Waza for the 20th anniversary of the language and the release of Ruby 2.0. If you weren't in the sold out crowd, not to worry. Information should flow free and experiences should be shared; in line with those concepts you can watch Matz's talk right here, then read about what's new in this version of Ruby and how to run it on Heroku.



With slides available on speakerdeck

20 years of simplicity, elegance, and programmer happiness

Heroku, since its founding, has been aligned with the key values of Ruby – simplicity, elegance, and programmer happiness. Heroku still believes in the power and flexibility of Ruby, and we've invested in the language by hiring **Yukihiro "Matz" Matsumoto**, **Koichi Sasada** and **Nobuyoshi Nakada**. We would like to thank them and the whole Ruby core team for making this release happen. Join us in celebrating Ruby's successes and in looking forward to the next twenty years by trying Ruby 2.0 on Heroku today.

Me!

Ruby apps are running using 1.8.7, you should upgrade. Ruby 1.8.7 is approaching End of Life (EOL) in three months on June 2013. EOL for Ruby 1.8.7 means no security or bug patches will be provided by the maintainers. Not upgrading means you're potentially opening up your application and your users to vulnerabilities. Don't wait till the final hour, upgrade now to be confident and secure.

Speed

Ruby 2.0 has a faster garbage collector and is **Copy on Write** friendly. Copy on Write or COW is an optimization that can reduce the memory footprint of a Ruby process when it is copied. Instead of allocating duplicate memory when a process is forked, COW allows multiple processes to share the same memory until one of the processes needs to modify a piece of information. Depending on the program, this optimization can dramatically reduce the amount of memory used to run multiple processes. Most Ruby programs are memory bound, so reducing your memory footprint with Ruby 2.0 may allow you to run more processes in fewer dynos.

*If you're not already running a concurrent backend consider trying the **Unicorn web server**.*

Features

In addition to running faster than 1.9.3, and having a smaller footprint, Ruby 2.0 has a number of new features added to the language including:

Mention about “Speed” of 2.0

Ruby 2.0 has a faster **garbage collector** and is **Copy on Write** friendly. Copy on
reduce the mem
copied. Instead
process is forked,
the same memory until one of the processes needs to modify a piece of information. Depending on the program, this optimization can dramatically reduce the amount of memory used to run multiple processes. Most Ruby programs are memory bound, so reducing your memory footprint with Ruby 2.0 may allow you to run more processes in fewer

Short summary: GC uses bitmap marking and CoW friendly

If you’re not alread
trying the Unicorn web server.

Short summary: Let’s try Unicorn!

Only mention about GC?

I DON’T work on GC!
People love GC performance

。+.。ヽ(*>∀<*)ノ。.+。

Let’s consider about GC/memory management!

Ruby 2.1 development

Ruby 2.1 internal features

- Internal hooks for memory management
- RGenGC: Restricted generational garbage collection

Today's topic

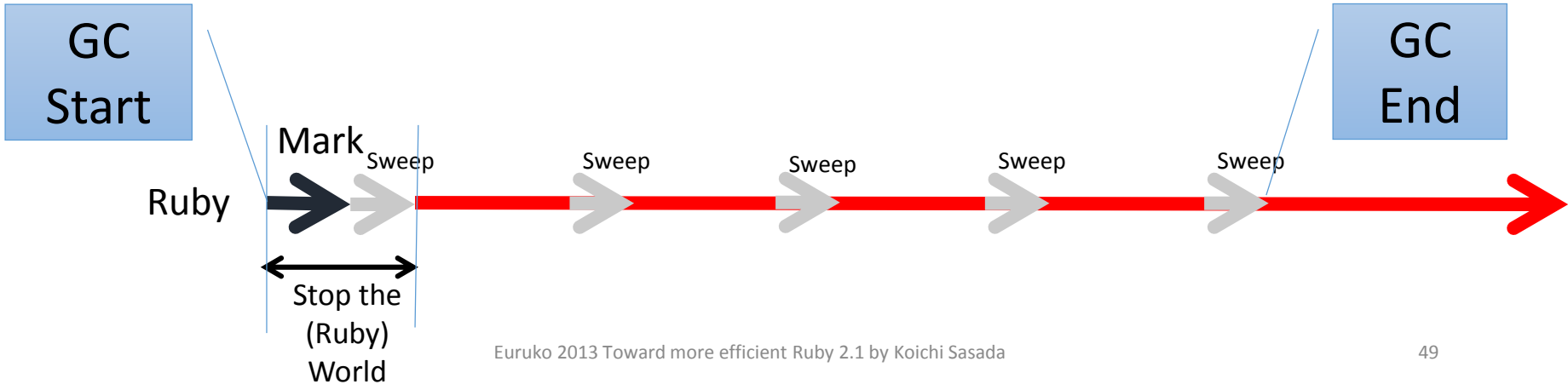
Ruby 2.1
Internal hooks for memory management

Internal hooks for memory management What's nice?

- You can collect more detailed analysis
- Examples
  - Collect object allocation site information
  - Collect usage of allocated objects
  - Measure GC performance from outside

Internal hooks for memory management

- Added events
  - RUBY_INTERNAL_EVENT_NEWOBJ
    - When object is created
  - RUBY_INTERNAL_EVENT_FREEOBJ
    - When object is freed
  - RUBY_INTERNAL_EVENT_GC_START
    - When GC is started
  - RUBY_INTERNAL_EVENT_GC_END
    - When GC is finished



Internal hooks for memory management *Caution*

- You can *NOT* trace these events using TracePoint (introduced from 2.0)
- You need to write C-ext to use them, because events are invoked during GC, etc

Internal hooks for memory management Sample features

- ObjectSpace. trace_object_allocations
  - Trace object allocation and record allocation-site
    - Record filename, line number, creator method's id and class
  - Usage:

```
ObjectSpace.trace_object_allocations{ # record only in the block
  o = Object.new
  file = ObjectSpace.allocation_sourcefile(o)   #=> __FILE__
  line = ObjectSpace.allocation_sourceline(o) #=> __LINE__ -2
}
```

  - Demonstration

Internal hooks for memory management Postponed job

- You may want to write hooks in Ruby

  → Use ‘Postponed job’
  - ‘Postponed jobs’ run at same timing as finalizers
  - Usage: rb_postponed_job_register(func, data)
  - `func(data)’ will be called at a safe-point

- See an sample code in “ext/objspace/gc_hooks.c”
  - ObjectSpace.after_gc_(start|end) = proc{GC.start}
  - Proc is called after GC

Ruby 2.1
RGenGC: new garbage collection

RGenGC: Summary

- RGenGC: Restricted Generational GC
  - New GC algorithm allows mixing “Write-barrier protected objects” and “WB unprotected objects”
  - **No** (mostly) **compatibility issue** with C-exts
- Inserting WBs gradually
  - We can concentrate WB insertion efforts for major objects and major methods
  - Now, **Array, String, Hash, Object, Numeric** objects are WB protected
    - Array, Hash, Object, String objects are very popular in Ruby
    - Array objects using **RARRAY_PTR() change to WB unprotected** objects (called as Shady objects), so existing codes still works.

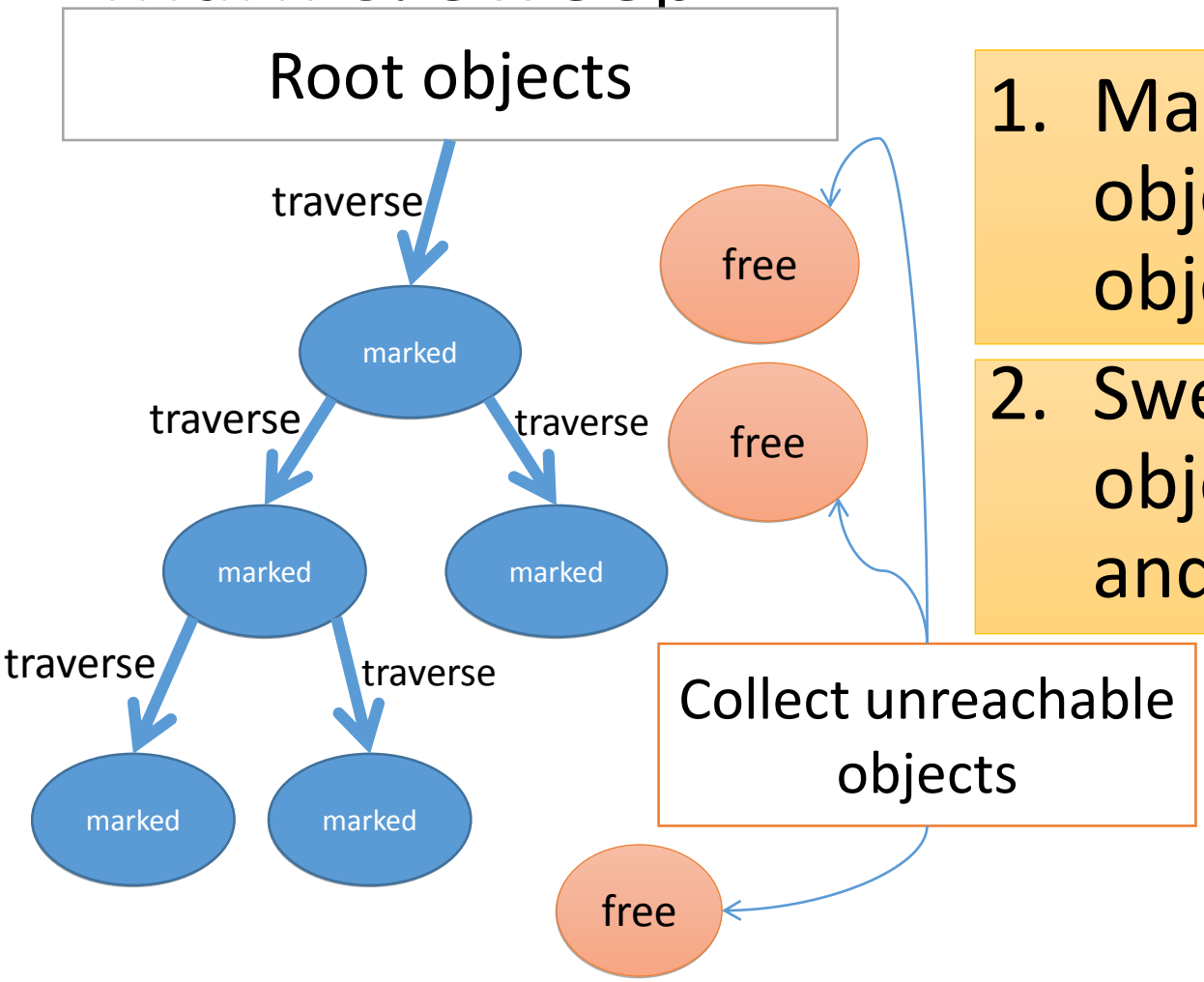
RGenGC: Agenda

- Background
  - Generational GC
  - Ruby's GC strategy
- Proposal: RGenGC
  - Separating into normal objects and shady objects
  - Shady objects at marking
  - Shade operation
- Implementation

RGenGC: Background
Current CRuby's GC

- Mark & Sweep
  - Conservative
  - Lazy sweep
  - Bitmap marking
  - Non-recursive marking
- C-friendly strategy
  - Don't need magical macros in C source codes
  - Many many C-extensions under this strategy

RGenGC: Background Mark & Sweep



1. Mark reachable objects from root objects
2. Sweep ***unmarked*** objects (collection and de-allocation)

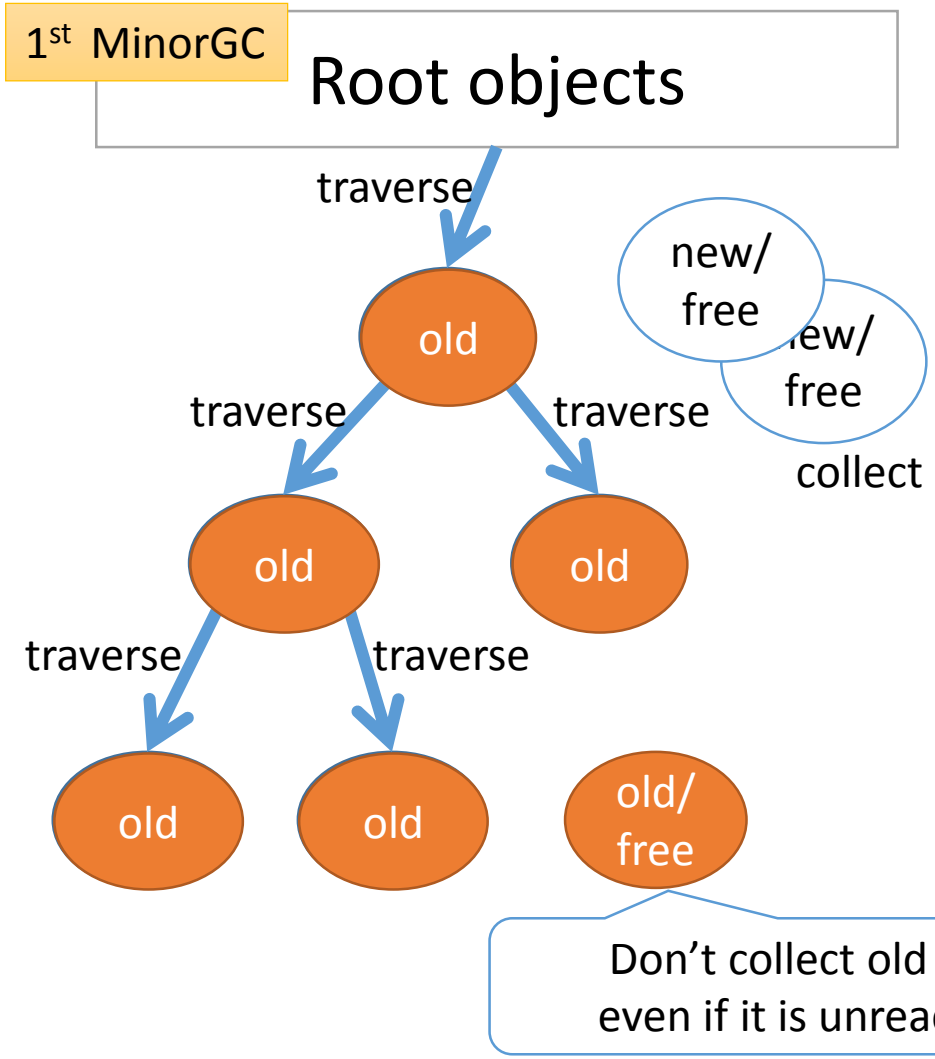
RGenGC: Background Generational GC (GenGC)

- Weak generational hypothesis: Most objects die young → Concentrating reclamation effort on the youngest objects
- Separate young generation and old generation
  - Create objects as young generation
  - Promote to old generation after surviving *n-th* GC
  - In CRuby, *n == 1* (after 1 GC, objects become old)
- Usually, GC on young space (minor GC)
- GC on both spaces if no memory (major/full GC)

RGenGC: Background Generational GC (GenGC)

- Minor GC and Major GC can use different GC algorithm
  - Popular combination
    → Minor GC: Copy GC, Major GC: M&S
  - **On the CRuby's:** both **Minor&Major GCs should be M&S** because CRuby's GC (and existing codes) based on conservative M&S algorithm

RGenGC: Background: GenGC [Minor M&S GC]

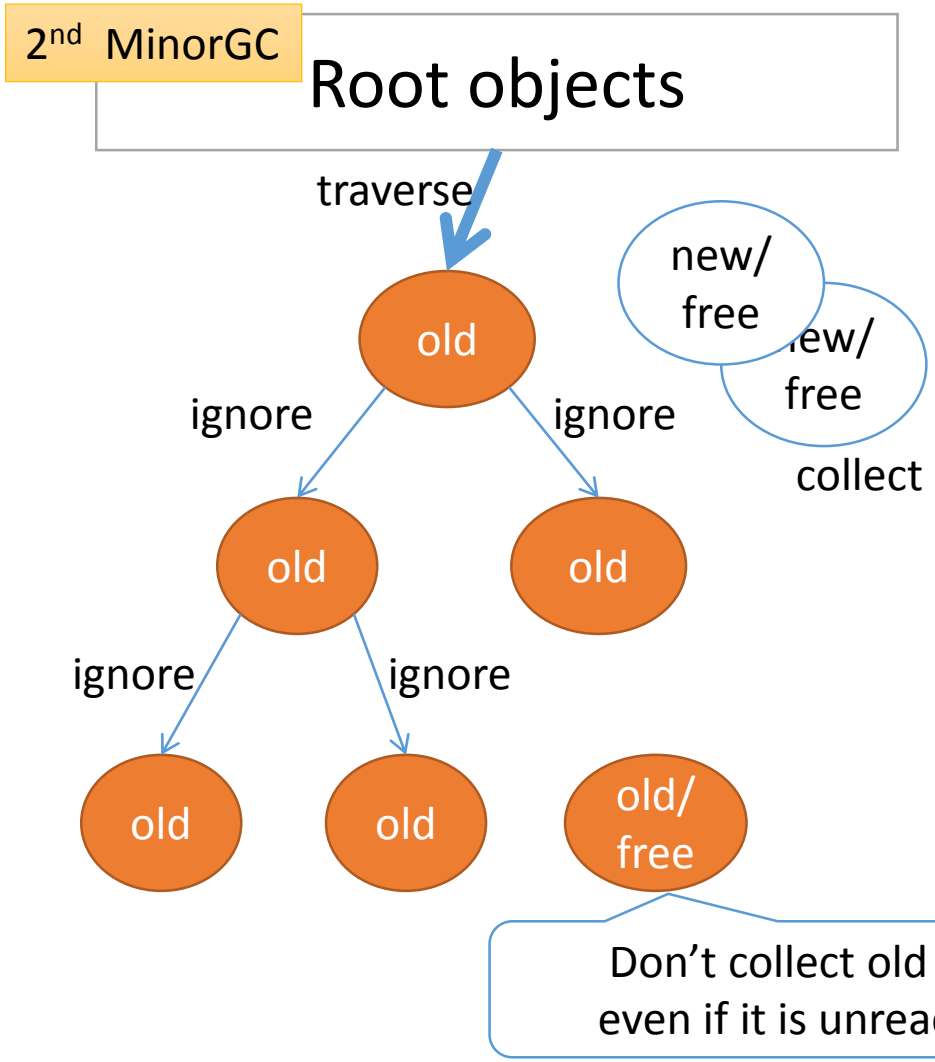


- Mark reachable objects from root objects.
  - Mark and **promote to old generation**
  - Stop traversing after old objects

→ Reduce mark overhead

- Sweep not (marked or old) objects
- Can't collect Some unreachable objects

RGenGC: Background: GenGC [Minor M&S GC]



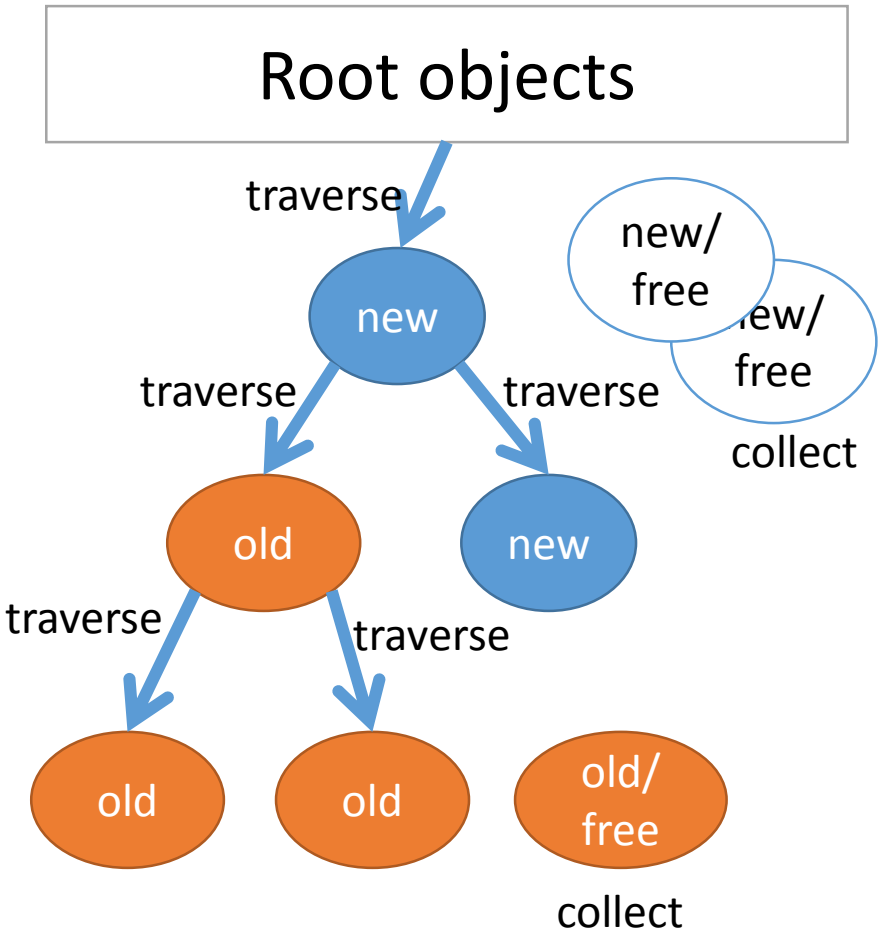
- Mark reachable objects from root objects.
  - Mark and **promote to old generation**
  - Stop traversing after old objects

→ Reduce mark overhead

- Sweep not (marked or old) objects

- Can't collect Some unreachable objects

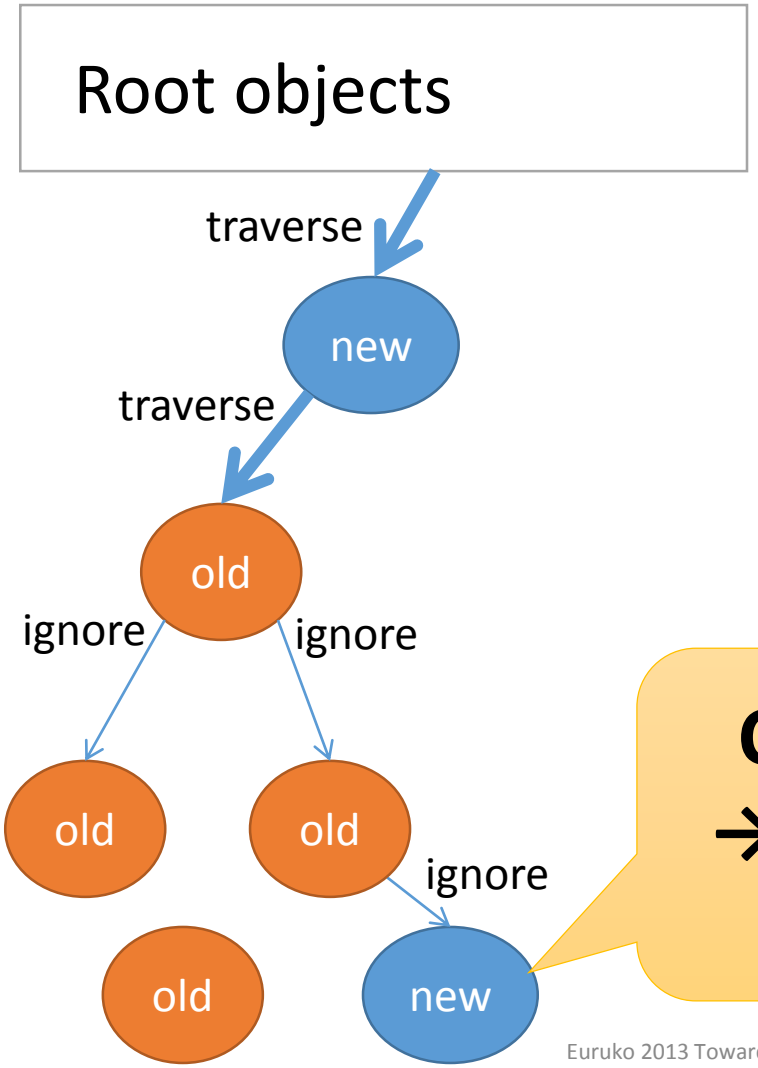
RGenGC: Background: GenGC [Major M&S GC]



- Normal M&S
- Mark reachable objects from root objects
  - Mark and **promote to old gen**
- Sweep unmarked objects

- *Sweep all unreachable (unused) objects*

RGenGC: Background: GenGC Problem: mark miss



- Old objects refer young objects

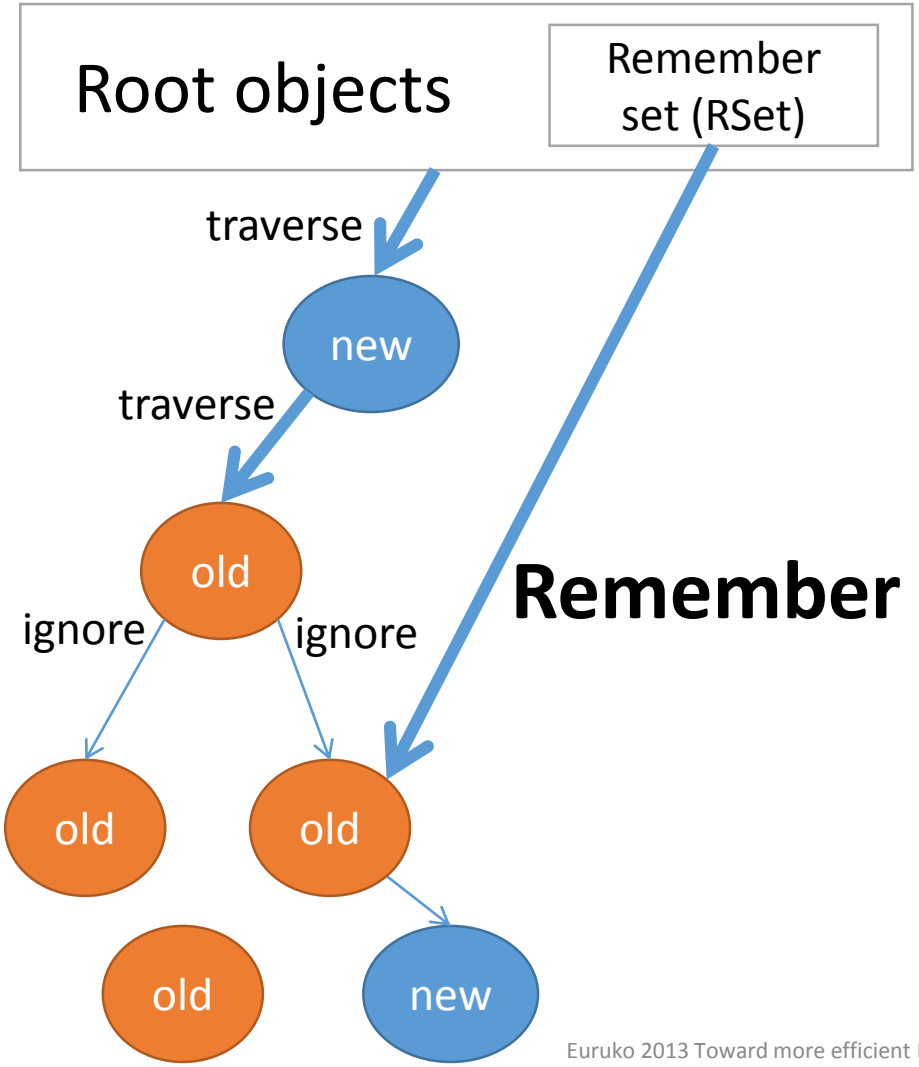
→ Ignore traversal of old object

→ Minor GC causes marking leak!!

- Because minor GC ignores referenced objects by old objects

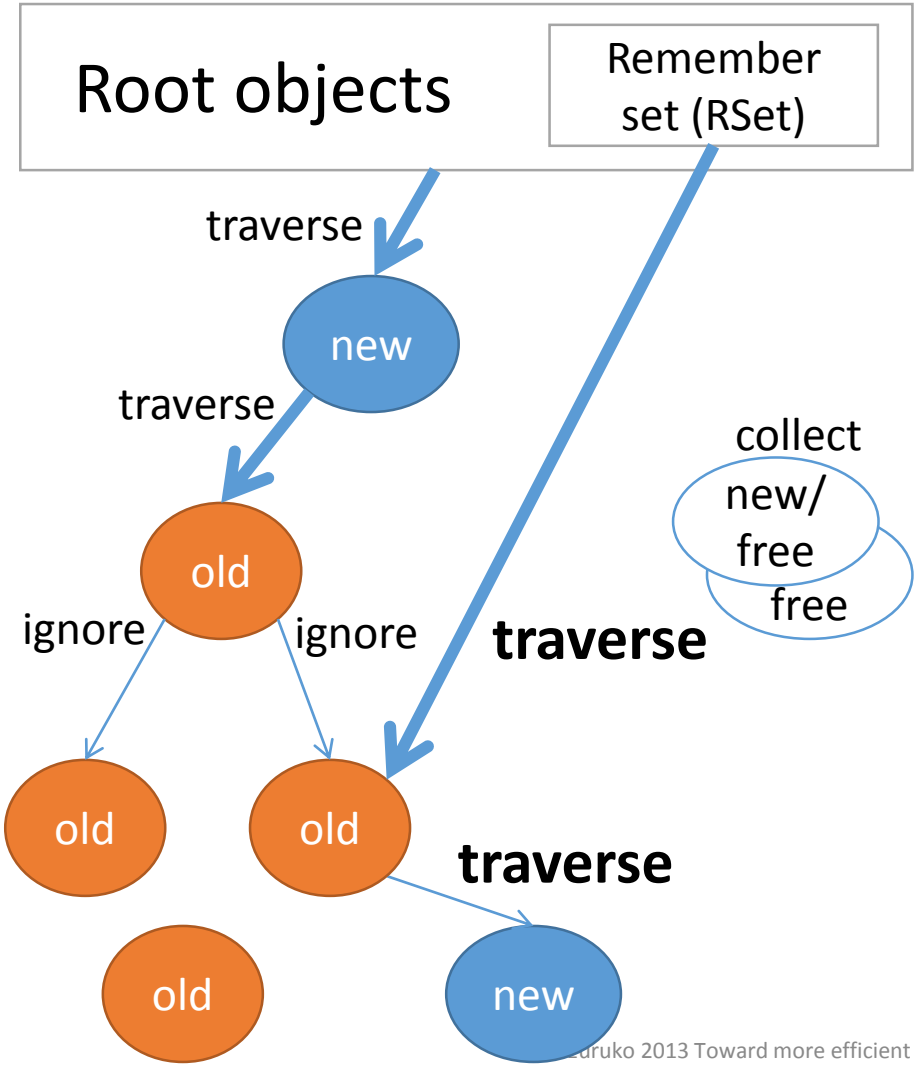
Can't mark new object!
→ Sweeping living object!
(Critical BUG)

RGenGC: Background: GenGC Introduce Remember set (Rset)



1. **Detect** creation of an [old->new] type reference
2. Add an [old object] into **Remember set (RSet)** if an old object refer new objects

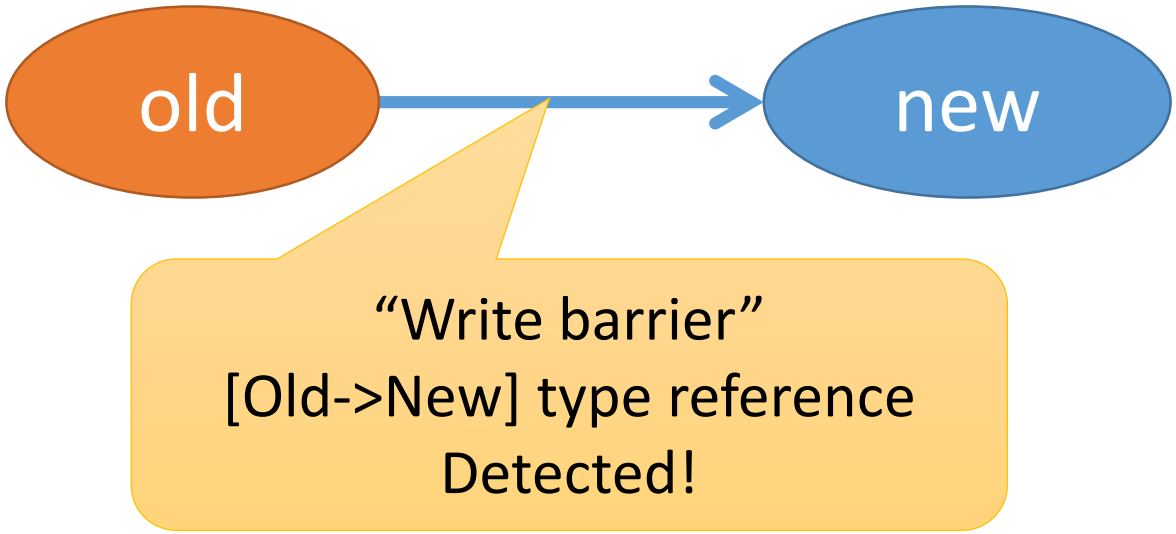
RGenGC: Background: GenGC [Minor M&S GC] w/ RSet



1. Mark reachable objects from root objects
   - **<u>Remembered objects are also root objects</u>**
2. Sweep not (marked or old) objects

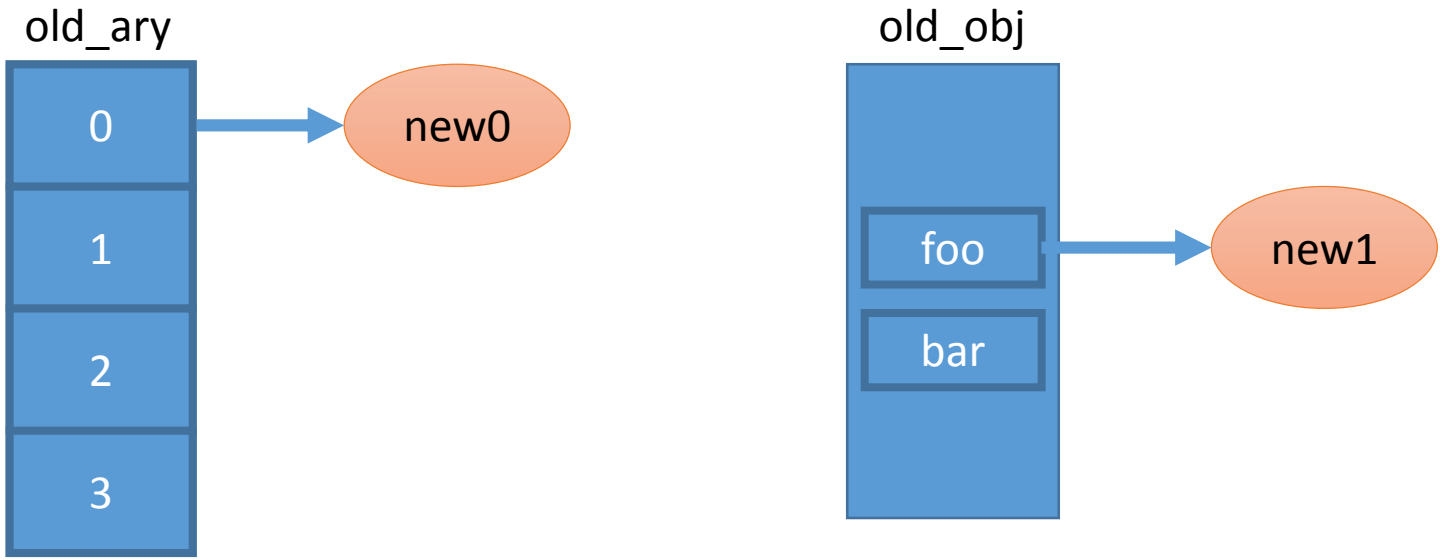
RGenGC: Background: GenGC Write barrier

- To detect [old→new] type references, we need to insert **“Write-barrier”** into interpreter for all “Write” operation



RGenGC: Background: GenGC Write barriers in Ruby

- Write barrier (WB) example in Ruby world
  - (Ruby) old_ary[0] = new0 # [old_ary → new0]
  - (Ruby) old_obj.foo = new1 # [old_obj → new1]



RGenGC: Background
Difficulty of inserting write barriers

- To introduce generational garbage collector, WBs are necessary to detect [old→new] type reference
- "Write-barrier miss" causes terrible failure
  1. WB miss
  2. Remember-set registration miss
  3. (minor GC) marking-miss
  4. **Collect live object → Terrible GC BUG!!**

RGenGC: Problem
Inserting WBs into C-extensions (C-exts)

- All of C-extensions need perfect Write-barriers
  - C-exts manipulate objects with Ruby's C API
  - C-level WBs are needed
- Problem: How to insert WBs into C-exts?
  - There are many WB required programs in C-exts
    - Example (C): RARRAY_PTR(old0)[0] = new1
    - Ruby C-API doesn't require WB before
  - CRuby interpreter itself also uses C-APIs
- How to deal with?
  - We can rewrite all of source code of CRuby interpreter to add WB, **with huge debugging effort!!**
  - We can't rewrite all of C-exts which are written by 3rd party

RGenGC: Problem
Inserting WBs into C-extensions (C-exts)

Two options

		Performance	Compatibility
1	Give up GenGC	Poor	Good (No problem)
2	GenGC with re-writing all of C exts	Good	Most of C-exts doesn't work

Current conservative choice

Trade-off of Speed and Compatibility

RGenGC: Related work on Ruby's GenGC

- Kiyama, et. al. GenGC for CRuby
  - Straightforward implementation for Ruby 1.6
  - Need WBs in correct places
  - High development cost
  - Can't keep compatibility → Drop all C-exts
- Nari, et.al longlife GC for CRuby
  - Introduce GenGC only for Node object
  - No compatibility issues because C-exts don't use node
  - Now CRuby doesn't use many number of node objects
  - High development cost (to guarantee WBs)

RGenGC:
Related work on Ruby's GenGC

- Make interpreter with other language infrastructures which have GC
  - JRuby, IronRuby
  - Can't keep compatibility with current C-exts
- Separate core heap and CRuby C-ext heap
  - High development cost

RGenGC: Challenge

- Trade-off of Speed and Compatibility
  - Can we achieve both **speed-up w/ GenGC** and **keeping compatibility?**

- Several possible approaches
  - Separate heaps into the WB world and non-WB world
    - Need to re-write whole of Ruby interpreter
    - Need huge development effort
  - WB auto-insertion
    - Modify C-compiler
    - Need huge development effort

RGenGC: Our approach

- Create **new generational GC algorithm** permits WB protected objects **AND** WB un-protected object in the same heap

RGenGC: Restricted Generational Garbage Collection

RGenGC: Invent 3rd option

		Performance	Compatibility
1	Give up GenGC	Poor	Good (No problem)
2	GenGC with re-writing all of C codes	Good	Most of C-exts doesn't work
3	Use new RGenGC	Good	Most of C-exts works!!

Ruby 2.1 choice

Breaking the trade off. You can praise us!!

RGenGC: Key idea

- Introduce **Shady object**
  - I use the word “Shady” as questionable, doubtful, …
  - Something feeling dark
  - 日陰者, in Japanese

RGenGC:
Key Idea

- Separate objects into two types
  - Normal Object: WB Protected
  - Shady Object: WB Unprotected

Shady: doubtful, questionable, ...

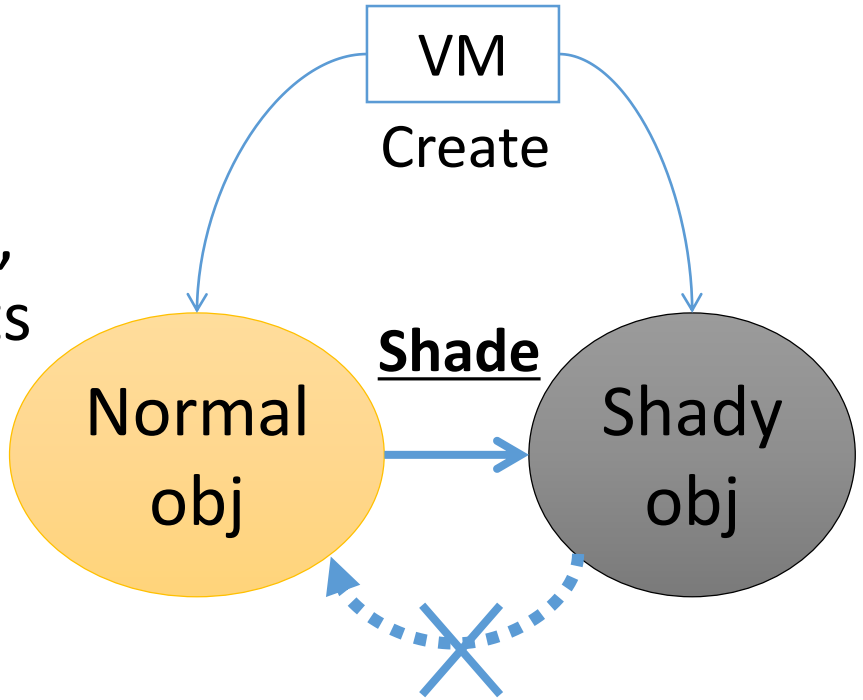
Normal
＼(^o^)／

Shady
(´･ω･`)

- We are not sure that a shady object points new objects or not
- Decide this type at creation time
  - A class care about WB → Normal object
  - A class don't care about WB → Shady object

RGenGC:
Key Idea

- Normal objects can be changed to Shady objects
  - "Shade operation"
  - C-exts don't care about WB, objects will be shady objects
  - Example
    - ptr = RARRAY_PTR(ary)
    - In this case, we can't insert WB for ptr operation, so VM shade "ary"

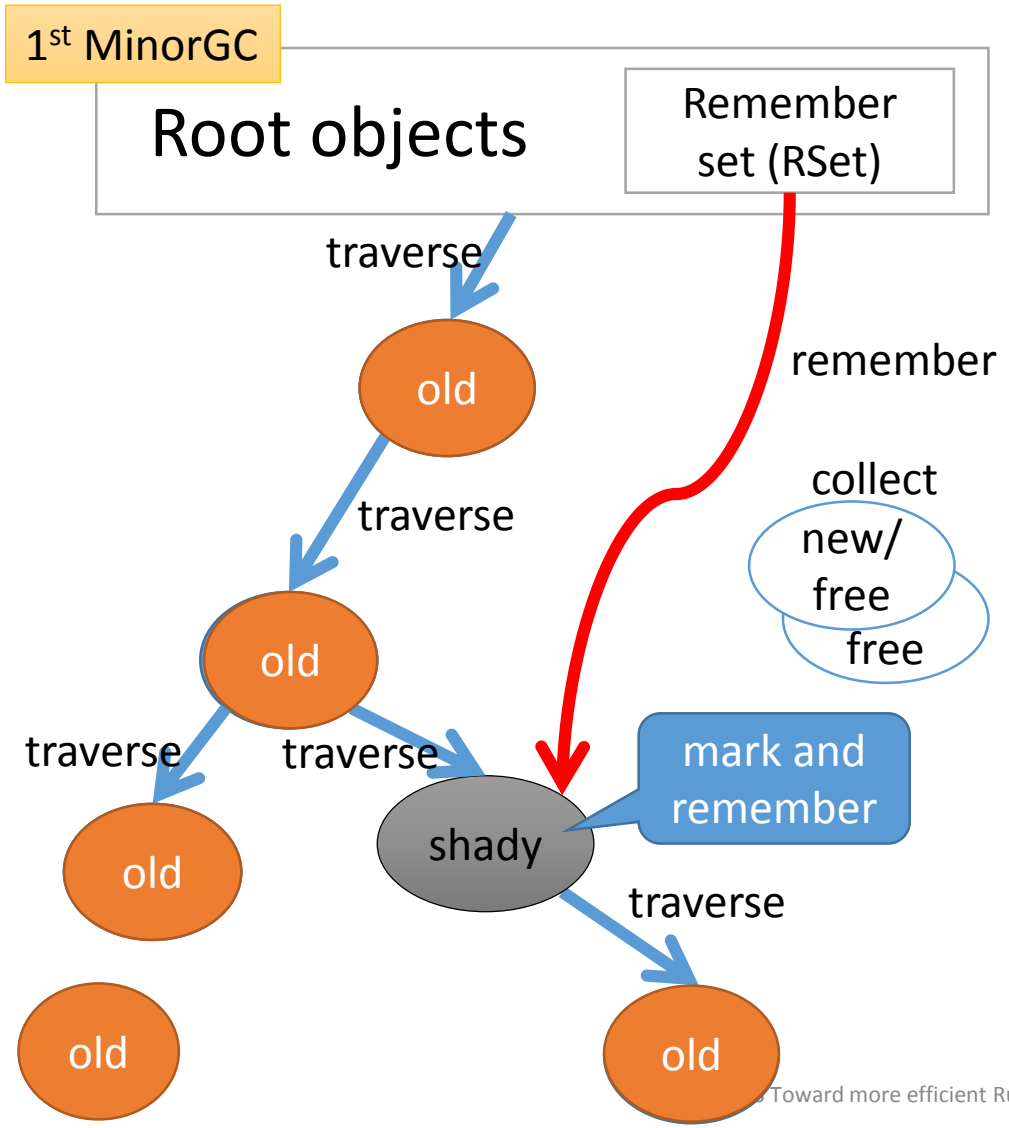


Now, Shady object **can't** change into normal object

RGenGC
Key Idea: Rule

- Treat “Shady objects” correctly
  - At Marking
    1. Don’t promote shady objects to old objects
    2. Remember shady objects pointed from old objects
  - At Shade operation for old normal objects
    1. Demote objects
    2. Remember shaded shady objects

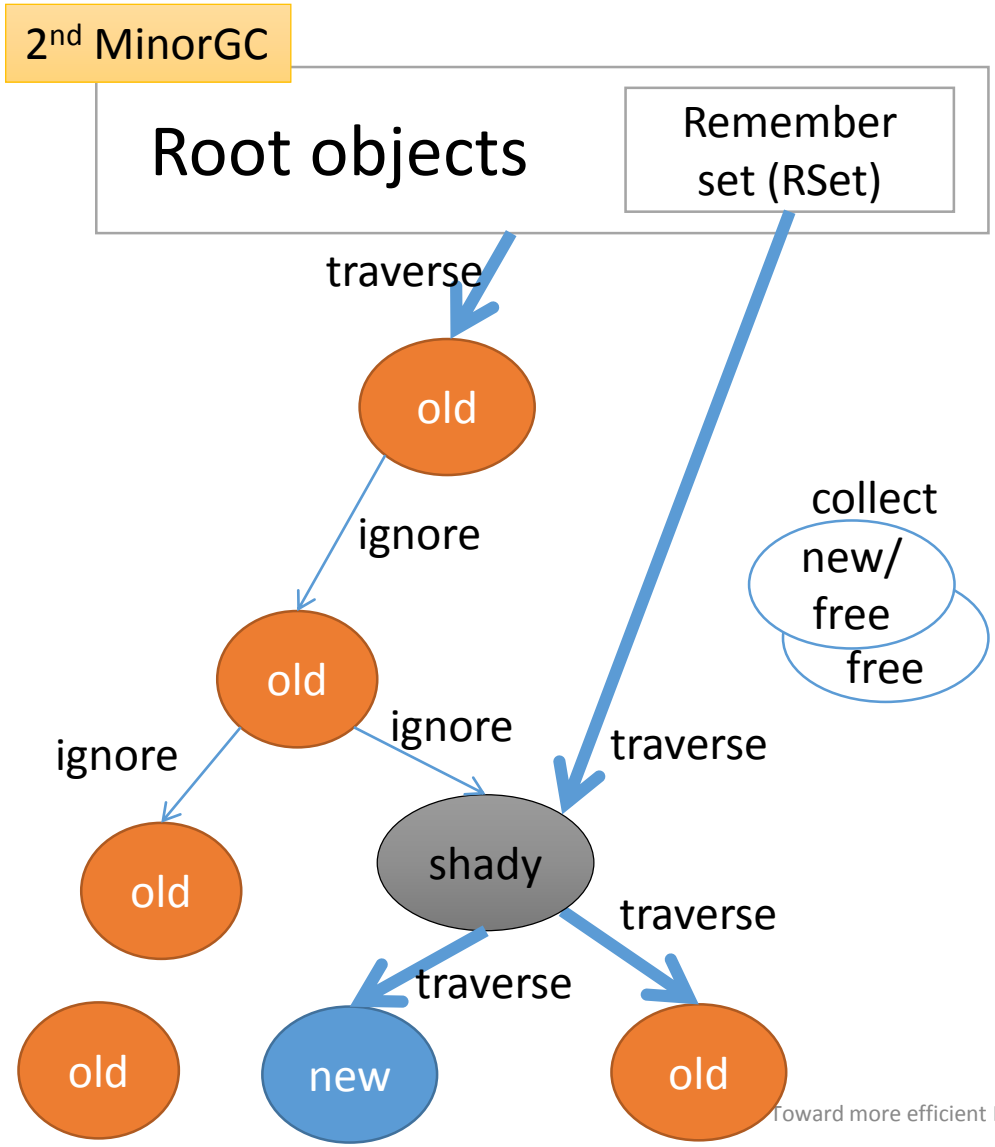
RGenGC
[Minor M&S GC w/Shady object]



- Mark reachable objects from root objects
  - Mark shady objects, and ***don't promote*** to old gen objects
  - If shady objects **pointed from old objects**, then **remember shady objects** by RSet.

    → Mark shady objects every minor GC!!

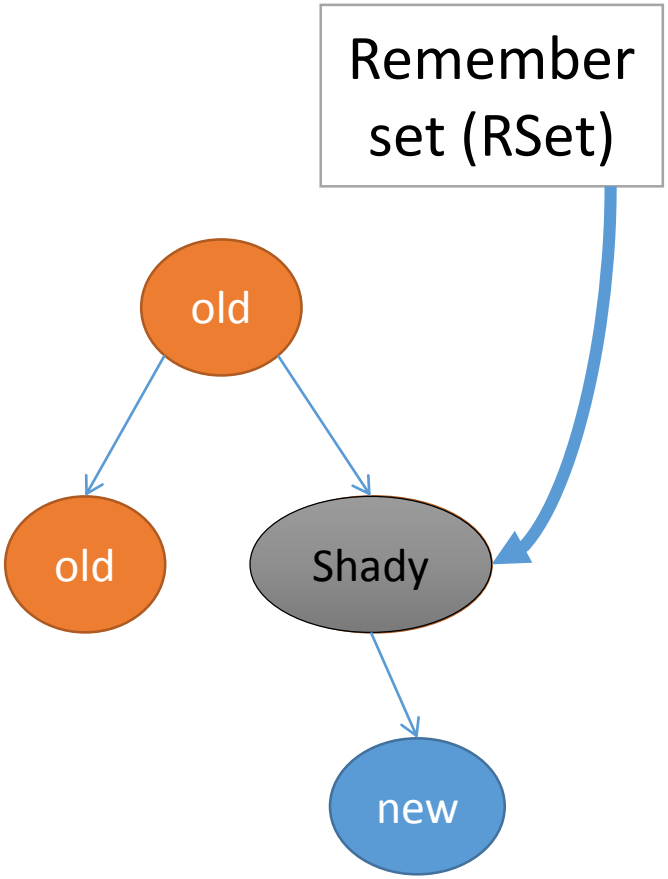
RGenGC
[Minor M&S GC w/Shady object]



- Mark reachable objects from root objects
  - Mark shady objects, and ***don't promote**** to old gen objects
  - If shady objects pointed from old objects, then remember shady objects by RSet.

    → Mark shady objects every minor GC!!

RGenGC
[Shade operation]



- Anytime Object can give up to keep write barriers

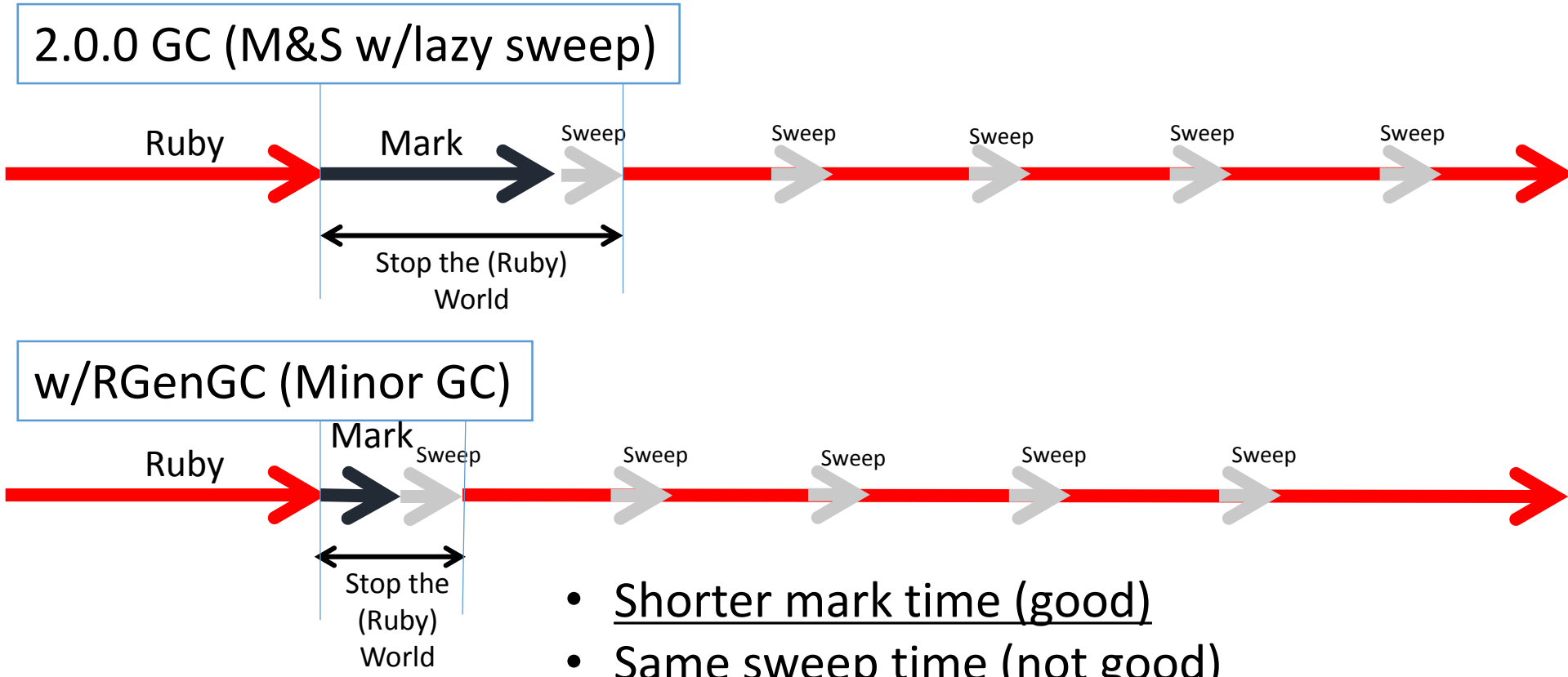
  → [Shade operation]

- Change old normal objects to shade objects
  - Example: RARRAY_PTR(ary)

  (1) Demote object (old → new)

  (2) Register it to Remember Set

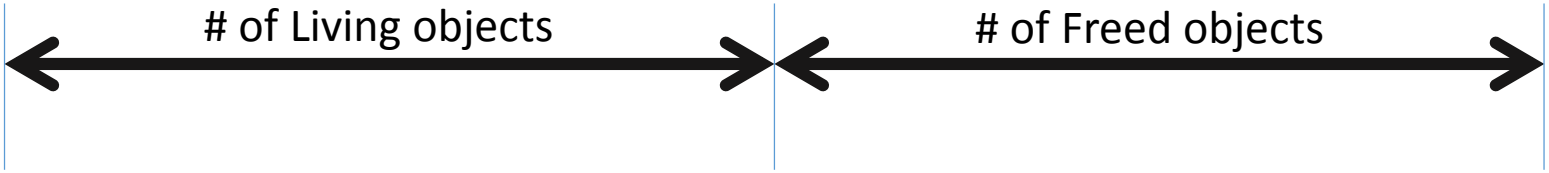
RGenGC Timing chart



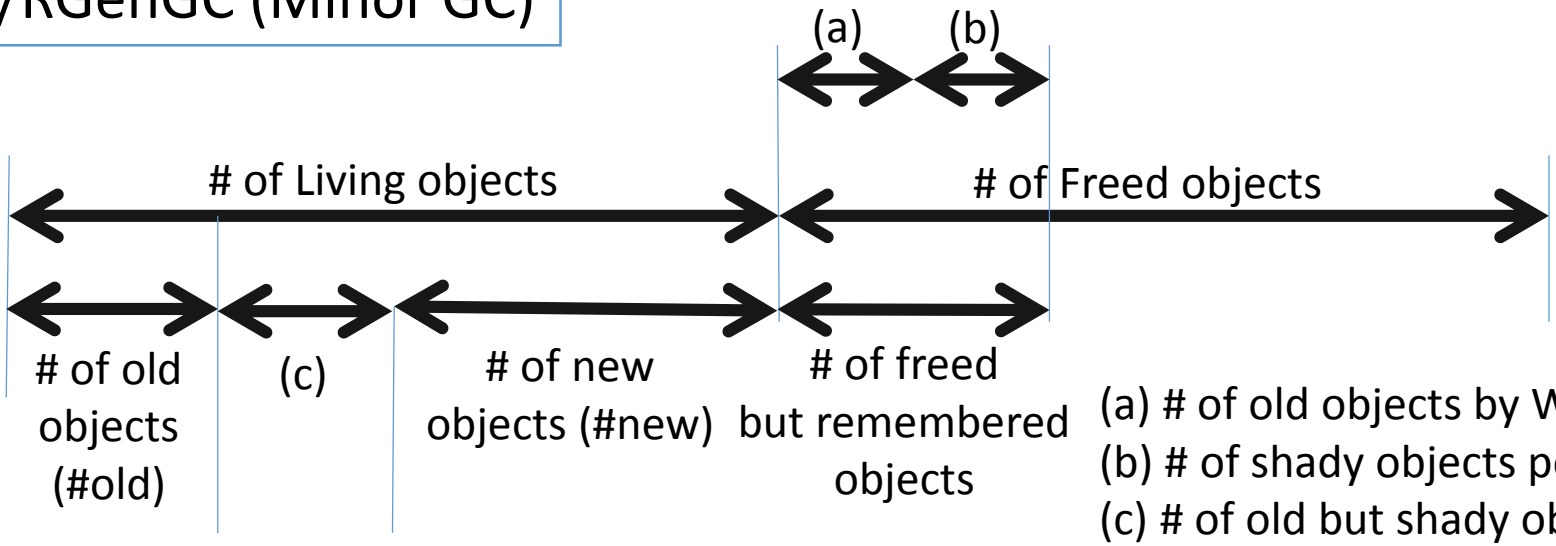
- Shorter mark time (good)
- Same sweep time (not good)
- (little) Longer execution time b/c WB (bad)

RGenGC
Number of objects

2.0.0 GC (M&S)



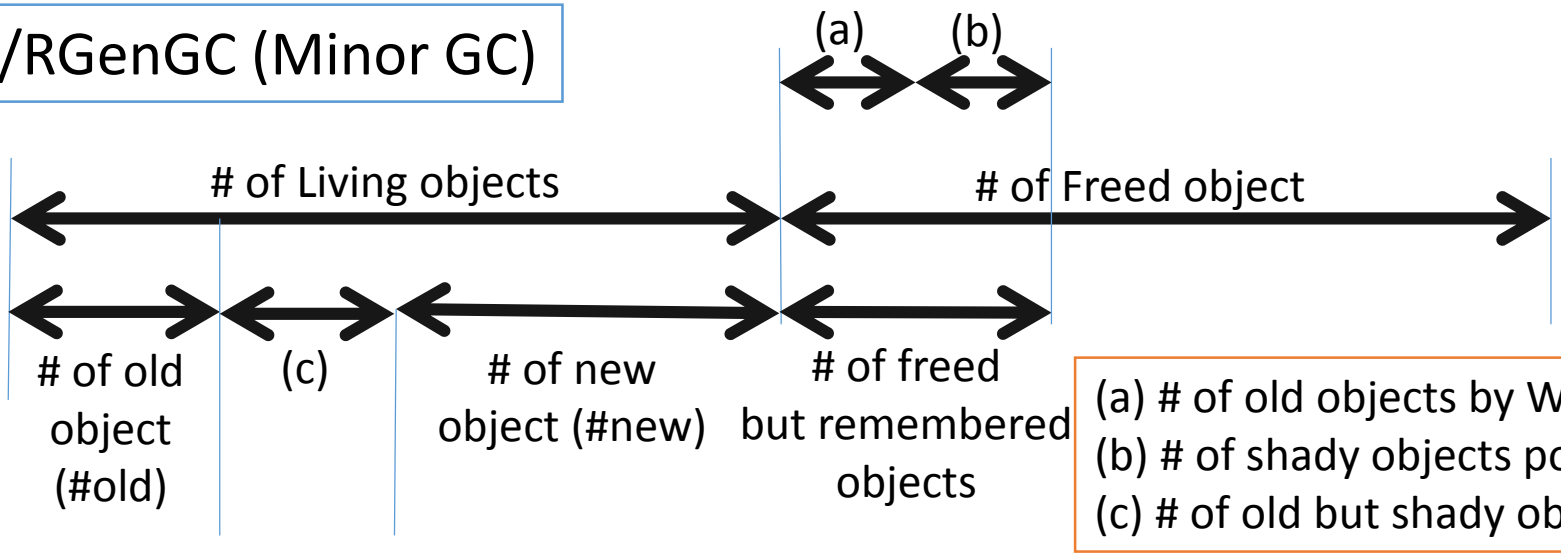
w/RGenGC (Minor GC)



(a) # of old objects by WB
(b) # of shady objects pointed by old
(c) # of old but shady objects

RGenGC
Number of objects

w/RGenGC (Minor GC)



(a) # of old objects by WB
(b) # of shady objects pointed by old
(c) # of old but shady objects

	Marking space	Number of unused, uncollected objs	Sweeping space
Mark&Swep GC	# of Living objects	0	Full heap
Traditional GenGC	#new + (a)	(a)	#new
RGenGC	#new + (a) + (b) + (c)	(a) + (b)	Full heap

RGenGC Discussion: Pros. and Cons.

- Pros.
  - Allow WB unprotected objects (shady objects)
    - **100% compatible** w/ existing extensions which don't care about WB
    - A part of CRuby interpreter which doesn't care about WB
  - **Inserting WBs step by step, and increase performance gradually**
    - We don't need to insert all WBs into interpreter core at a time
    - We can concentrate into popular (effective) classes/methods.
    - We can ignore minor classes/methods.
  - Simple algorithm, easy to develop (already done!)

RGenGC
Discussion: Pros. and Cons.

- Cons.
  - Increasing “unused, but not collected objects until full/major GC
    - Remembered normal objects (caused by traditional GenGC algorithm)
    - Remembered shady objects (caused by RGenGC algorithm)
  - WB insertion bugs (GC development issue)
    - RGenGC permit shady objects, but sunny objects need correct/perfect WBs. But inserting correct/perfect WBs is difficult.
    - This issue is out of scope. We have another idea against this problem (out of scope).
  - Can’t reduce Sweeping time
    - But many (and easy) well-known techniques to reduce sweeping time (out of scope).

RGenGC Implementation: WB support status

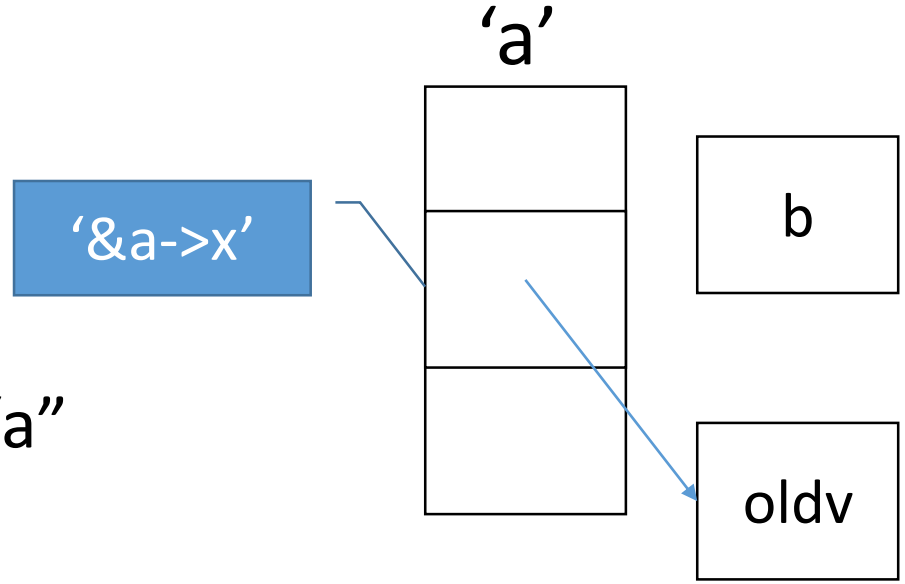
Type name	Status	Comment
T_OBJECT	Supported	
T_CLASS	Supported	Possible to change into shady
T_ICLASS	Supported	Possible to change into shady
T_MODULE	Supported	Possible to change into shady
T_FLOAT	Supported	
T_STRING	Supported	
T_REGEXP	Supported	
T_ARRAY	Supported	Possible to change into shady / more efforts are needed
T_HASH	Supported	Possible to change into shady
T_STRUCT	Supported	
T_BIGNUM	Supported	
T_FILE	Unsupported	
T_DATA	Supported	Only InstructionSequence objects are supported
T_MATCH	Unsupported	Most of MatchData objects are short-lived
T_RATIONAL	Supported	
T_COMPLEX	Supported	
T_NODE	Unsupported	Most of Node objects are short-lived

RGenGC Implementation

- Introduce two flags into RBasic
  - FL_KEEP_WB: WB protected or not protected
    - 0 → unprotected → Shady object
    - 1 → protected → Sunny object
    - Usage: NEWOBJ_OF(ary, struct RArray, klass, T_ARRAY | **FL_KEEP_WB**);
  - FL_OLDGEN: Young gen or Old gen?
    - 0 → Young gen
    - 1 → Old gen
    - Don’t need to touch by user program
- Remember set is represented by bitmaps
  - Same as marking bitmap
  - heap_slot::rememberset_bits
  - Traverse all object area with this bitmap at first

RGenGC Implementation: WB operation API

- OBJ_WRITE(a, &a->x, b)
  - Declare ‘a’ aggregates ‘b’
  - **Write:** *&a->x = b
  - Write barrier
  - OBJ_WRITE(a, b) returns “a”



- OBJ_WRITTEN(a, oldv, b)
  - Declare ‘a’ aggregates ‘b’ and old value is ‘oldv’
  - Non-write operation
  - Write barrier

RGenGC Implementation: WB operation API

- T_ARRAY
  - **RARRAY_PTR(ary) causes shade operation**
    - Can't get RGenGC performance improvement
    - But works well ☺
- Instead of RARRAY_PTR(ary), use alternatives
  - **RARRAY_AREF(ary, n)** → RARRAY_PTR(ary)[n]
  - **RARRAY_ASET(ary, n, obj)** → RARRAY_PTR(ary)[n] = obj w/ Write-barrier
  - **RARRAY_PTR_USE(ary, ptrname, {...block...})**
    - Only in block, pointers can be accessed by `ptrname' variable (VALUE*).
    - **Programmers need to insert collect WBs (miss causes BUG)**.

RGenGC Incompatibility

- Make RBasic::klass “const”
  - Need WBs for a reference from an object to a klass.
  - Only few cases (zero-clear and restore it)
  - Provide alternative APIs
    - Now, RBASIC_SET_CLASS(obj, klass) and RBASIC_CLEAR_CLASS(obj) is added. But they should be internal APIs (removed soon).
    - rb_obj_hide() and rb_obj_reveal() is provided.

RGenGC Implementation

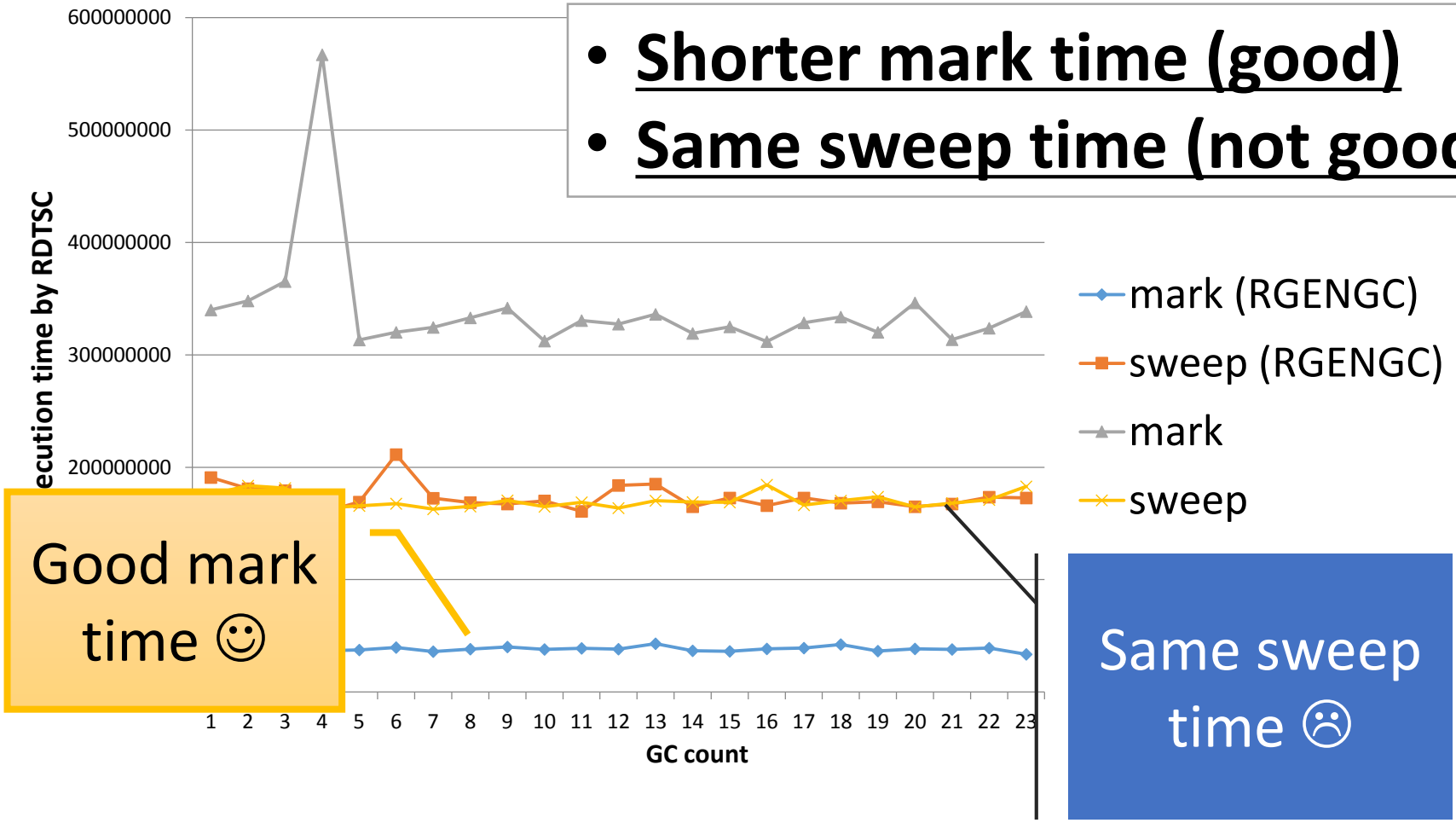
- RGENGC_CHECK_MODE in gc.c
  - 1: Enable assertions
  - 2: Enable “WB checking” mode
- WB checking mode
  - (1) do minor GC
  - (2) do major/full GC
  - (3) compare result with (1) and (2)
    - If living objects in (2) but not living in (1) it should be BUG!!
  - Not a perfect (implementation limitation), but a good method to detect bugs

RGenGC
Performance evaluation

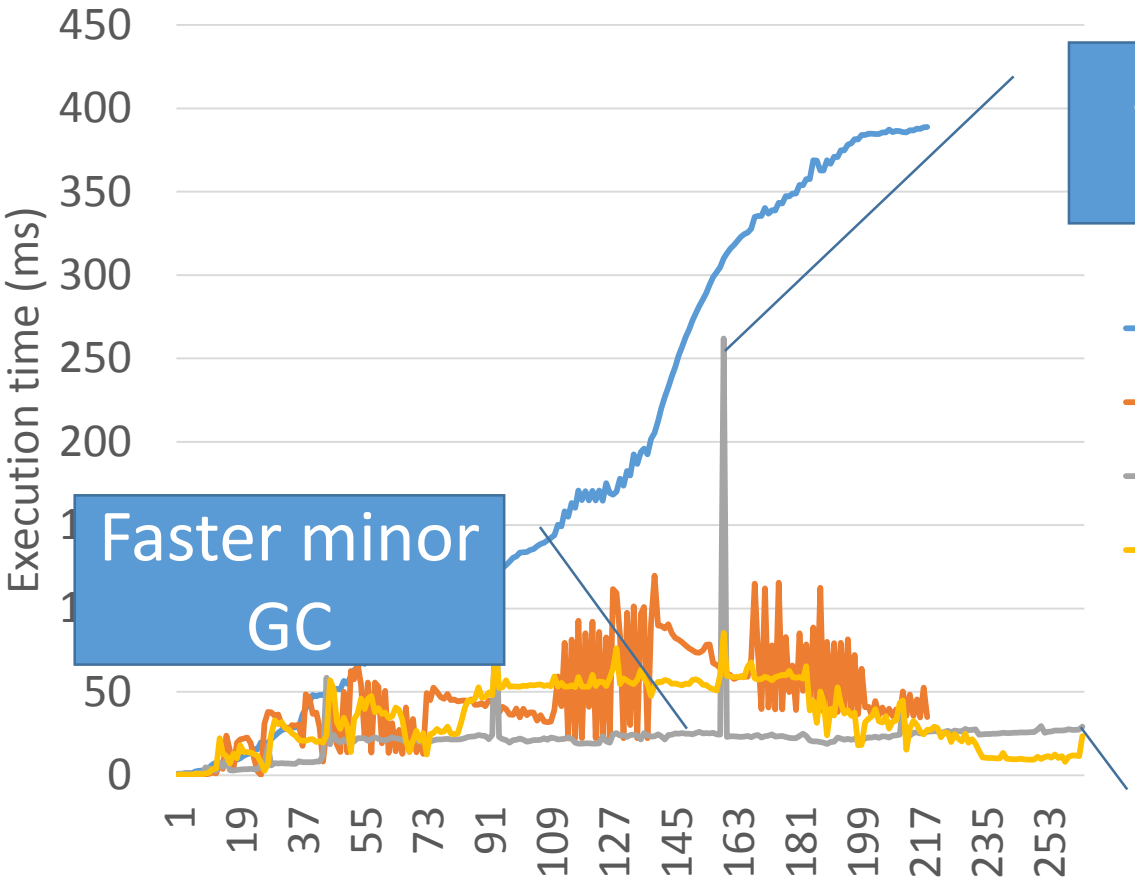
- Ideal micro-benchmark for RGenGC
  - Create many old objects at first
  - Many new objects (many minor GC, no major GC)
- RDoc
  - Same RDoc generation as Ruby's trunk

RGenGC
Performance evaluation (micro)

- **Shorter mark time (good)**
- **Same sweep time (not good)**



RGenGC
Performance evaluation (RDoc)



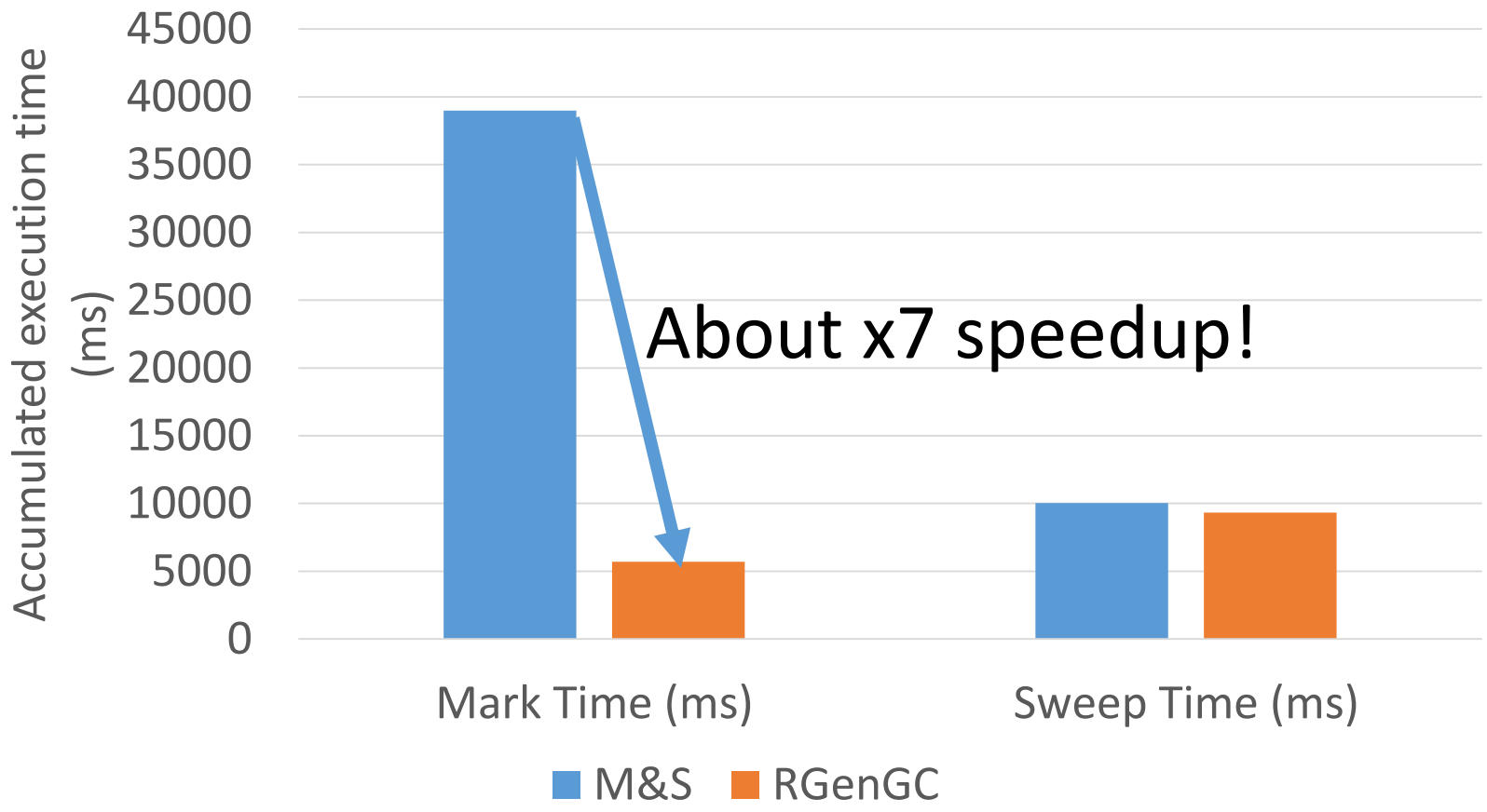
Several major/full GC peaks

Faster minor GC

Total GC count is different

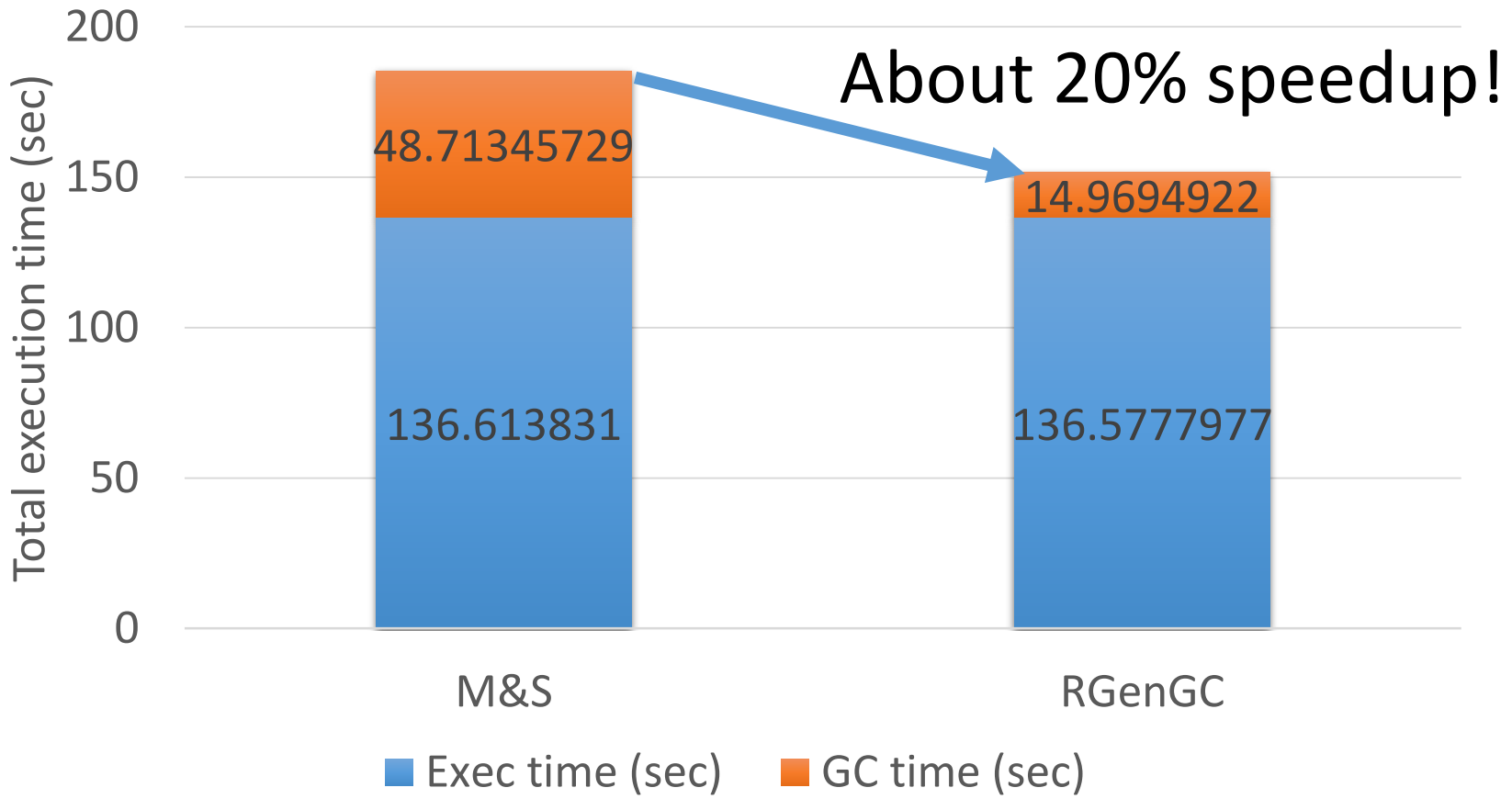
* Disabled laz ctly.

RGenGC
Performance evaluation (RDoc)



* Disabled lazy sweep to measure correctly.

RGenGC
Performance evaluation (RDoc)



* Disabled lazy sweep to measure correctly.

RGenGC: Summary

- RGenGC: Restricted Generational GC
  - New GC algorithm allow mixing “Write-barrier protected objects” and “WB unprotected objects”
  - **No** (mostly) **compatibility issue** with C-exts
- Inserting WBs gradually
  - We can concentrate WB insertion efforts for major objects and major methods

RGenGC
Future work

- Minor GC / Major GC timing tuning
  - Too many major GC → slow down
  - Too few major GC → memory consumption issue
- Inserting WBs w/ application profiling
  - Profiling system
  - Benchmark programs
- Debugging/Detecting system for WBs bugs
- Improve sweeping performance

Ruby 2.1
Other internal features

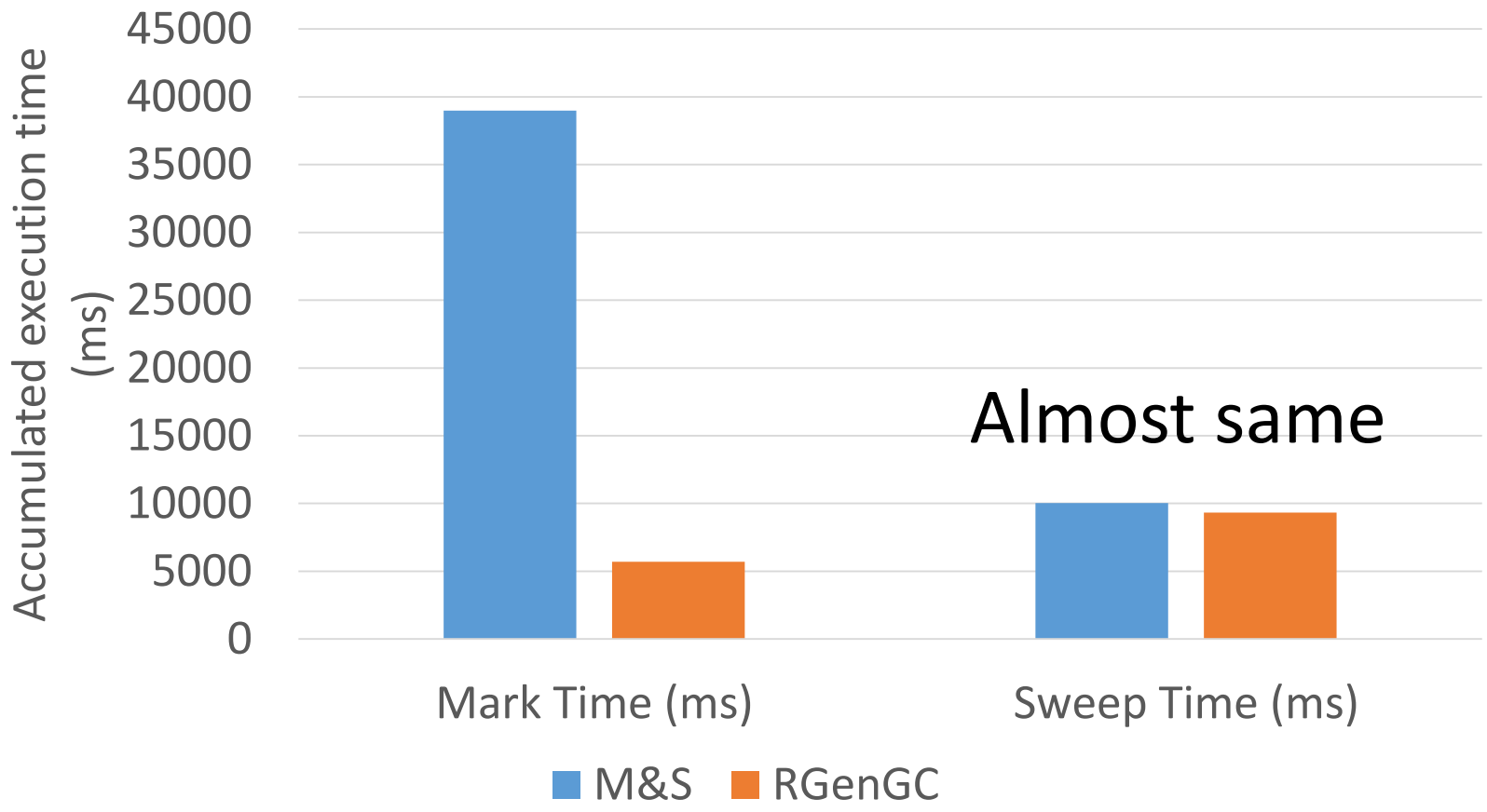
Ruby 2.1 expected “internal” features

- Parallel sweeping
- Sophisticated inline cache invalidation mechanism
- Memory efficient string management & Symbol GC
- Fine-grain memory protection to detect WB insertion miss
- Signal thread
- More efficient inter-process migration technique
- JIT compilation for small part of Ruby code
- Introduce fastpath C-methods
- Inlined Proc.call invocation
- AOT Compiler and extending “require” behavior
- Useful debugger

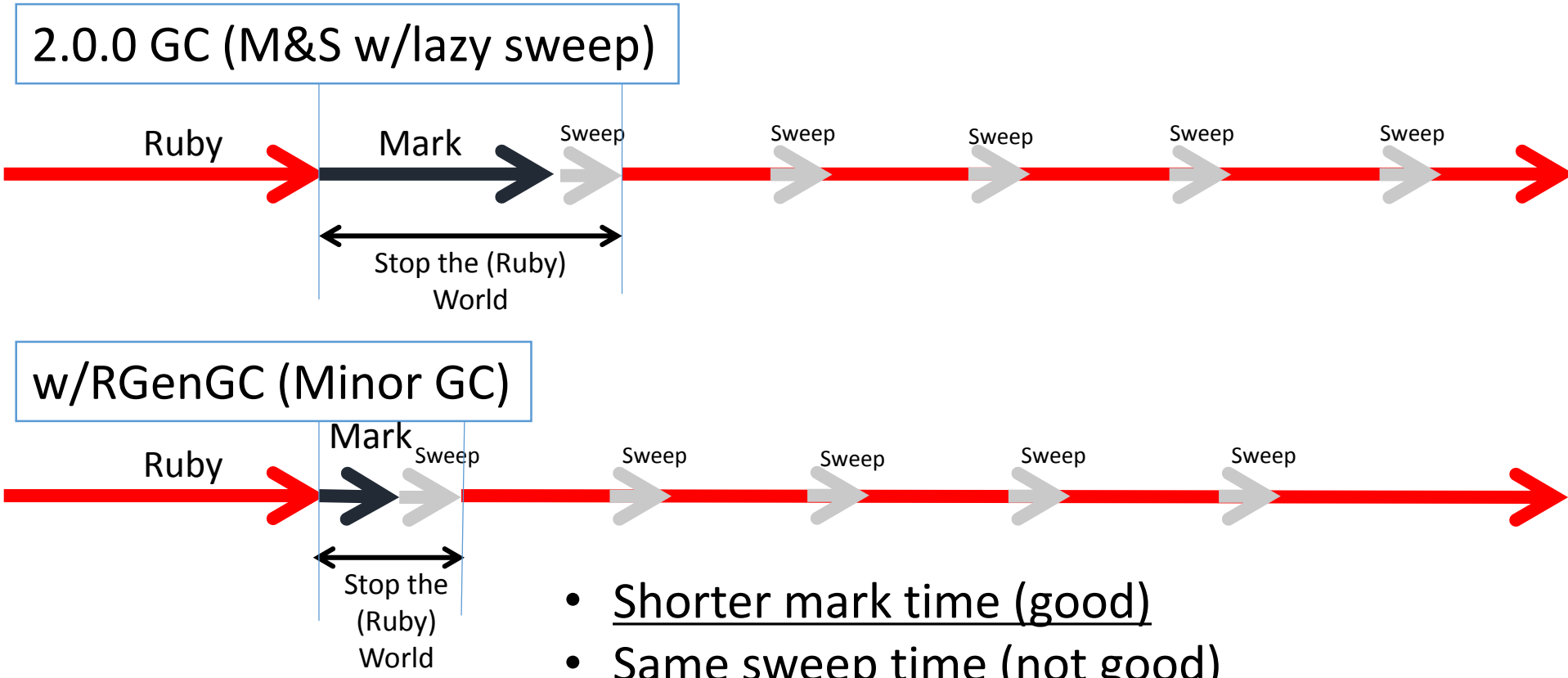
Parallel sweeping
Background

- RGenGC improve performance only for “marking” phase
- RGenGC doesn’t improve “sweeping phase” performance

Parallel sweeping
Background (revisit Rdoc evaluation)

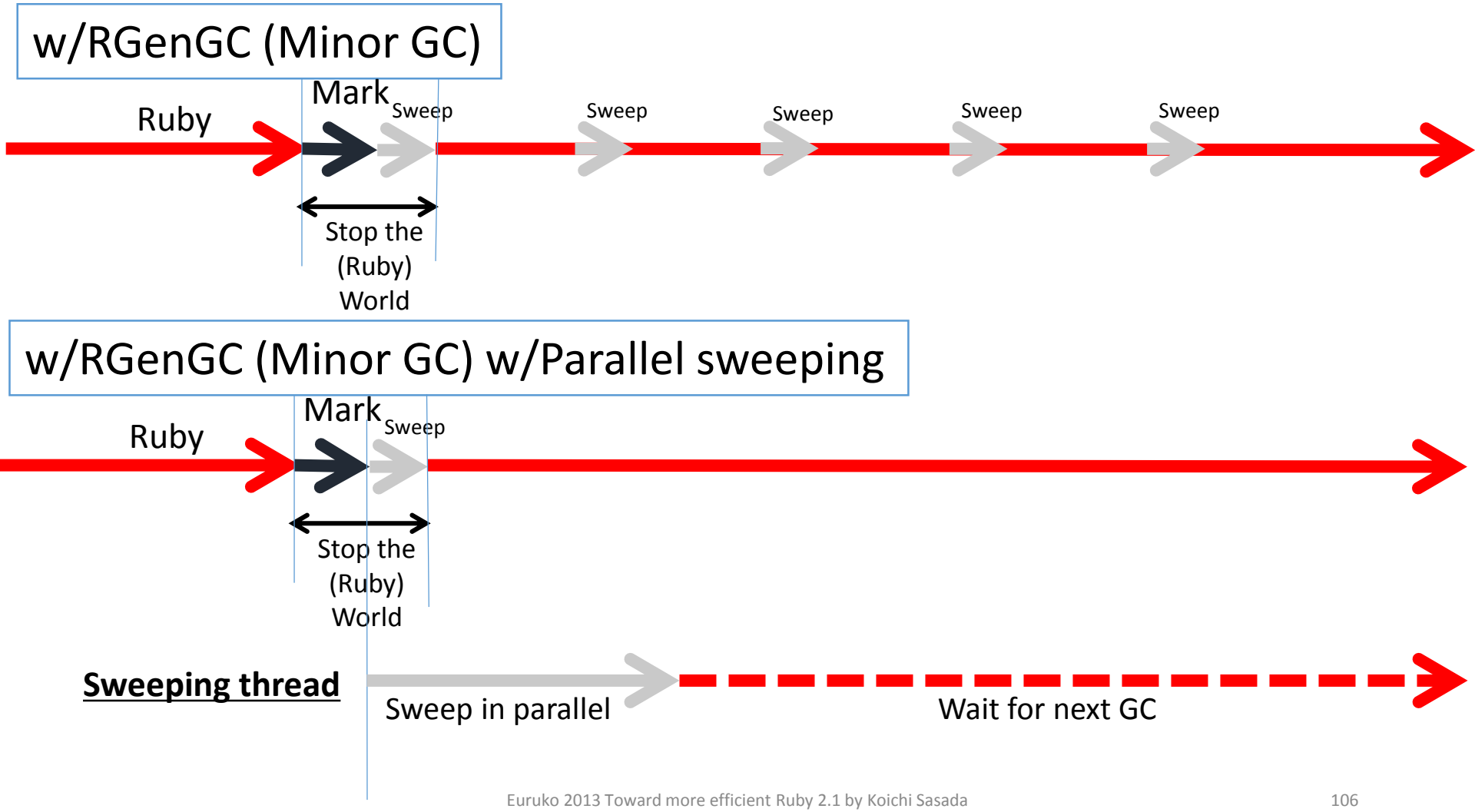


Parallel sweeping
Background (revisit RGenGC Timing chart)



- Shorter mark time (good)
- Same sweep time (not good)
- (little) Longer execution time b/c WB (bad)

Parallel sweeping
Introduce sweeping threads (ideal)



Parallel sweeping
Ideal

- Hide most of sweeping time

Parallel sweeping
Real

- Increase synchronization cost
- Increase program complexity

- Our preliminary evaluation (implemented in one night, buggy one) doesn't show good score

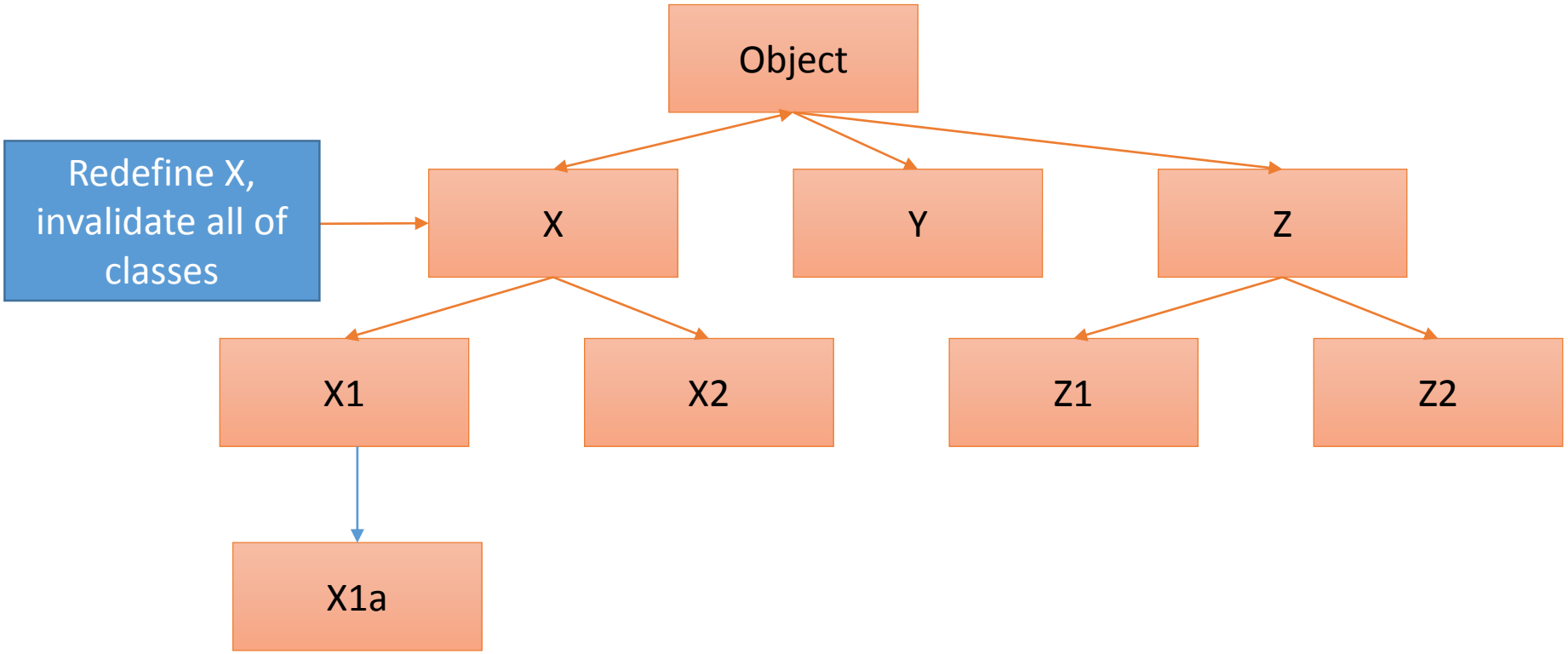
- To be continued…

Sophisticated inline cache invalidation mechanism

- From Ruby 1.9 (YARV), inline cache technique is used in several codes
  - Inline method caching ← Huge opportunity
  - Constant lookup
  - …
- Cache invalidation with only one variable “global_state_version”
- Invalidate inline cache, other non-related inline caches are also invalidated

Sophisticated inline cache invalidation mechanism

- Invalidate all classes' method cache



Sophisticated inline cache invalidation mechanism

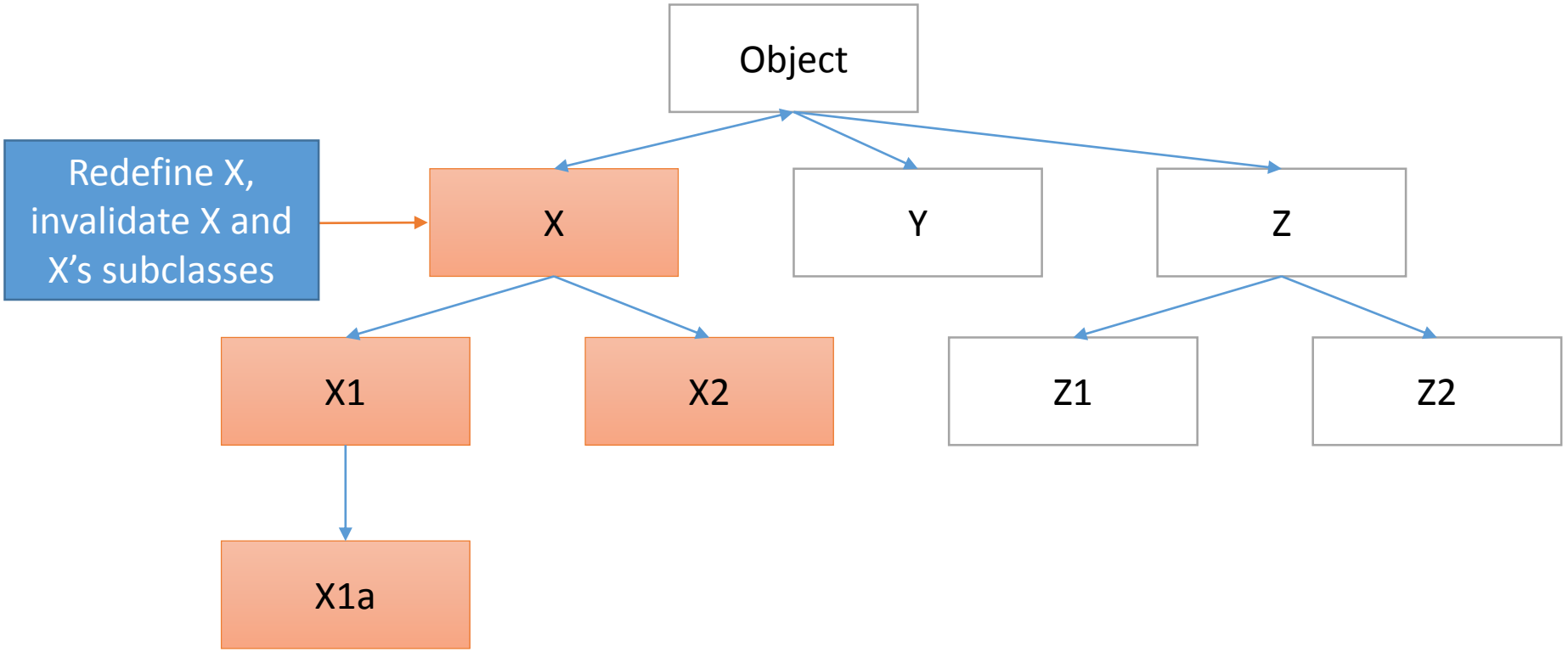
“This patch adds class hierarchy method caching to CRuby. This is the algorithm used by JRuby and Rubinius.”

[ruby-core:55053] [ruby-trunk - Feature #8426][Open] Implement class hierarchy method caching

by Charlie Somerville

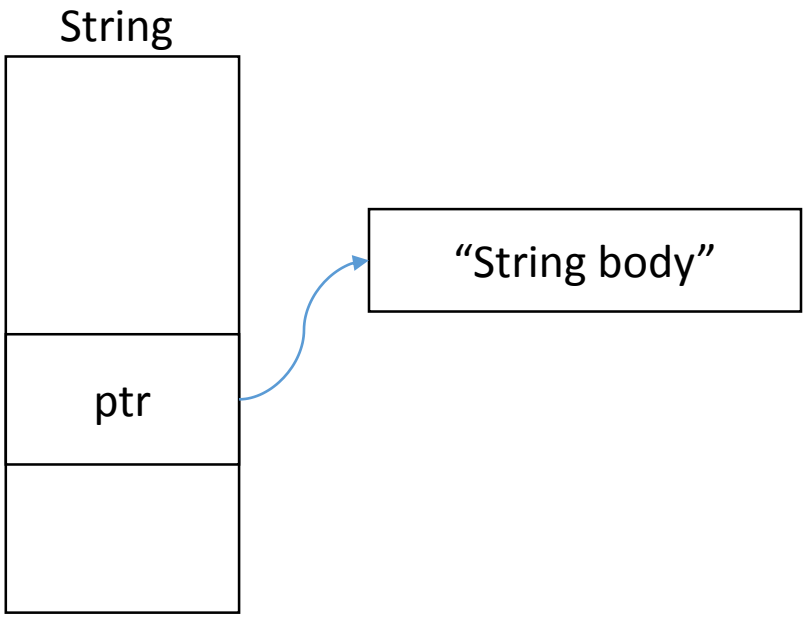
Sophisticated inline cache invalidation mechanism

- Invalid only sub-classes under effective class



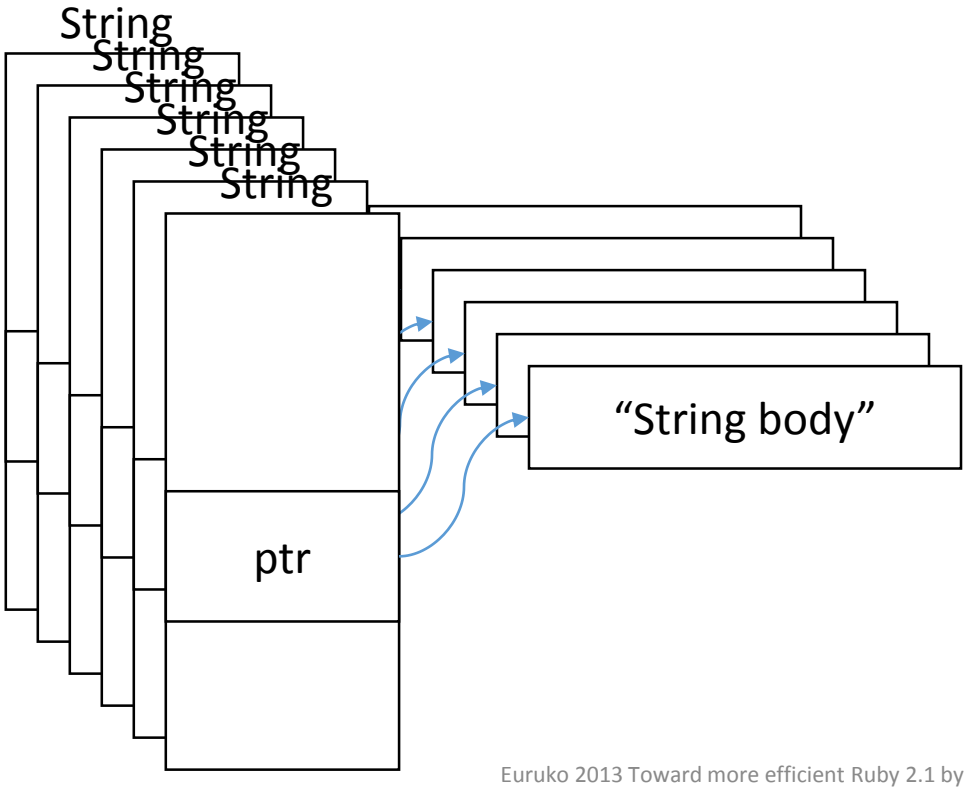
Memory efficient string management

- Each string has their string body (space acquired by malloc())



Memory efficient string management

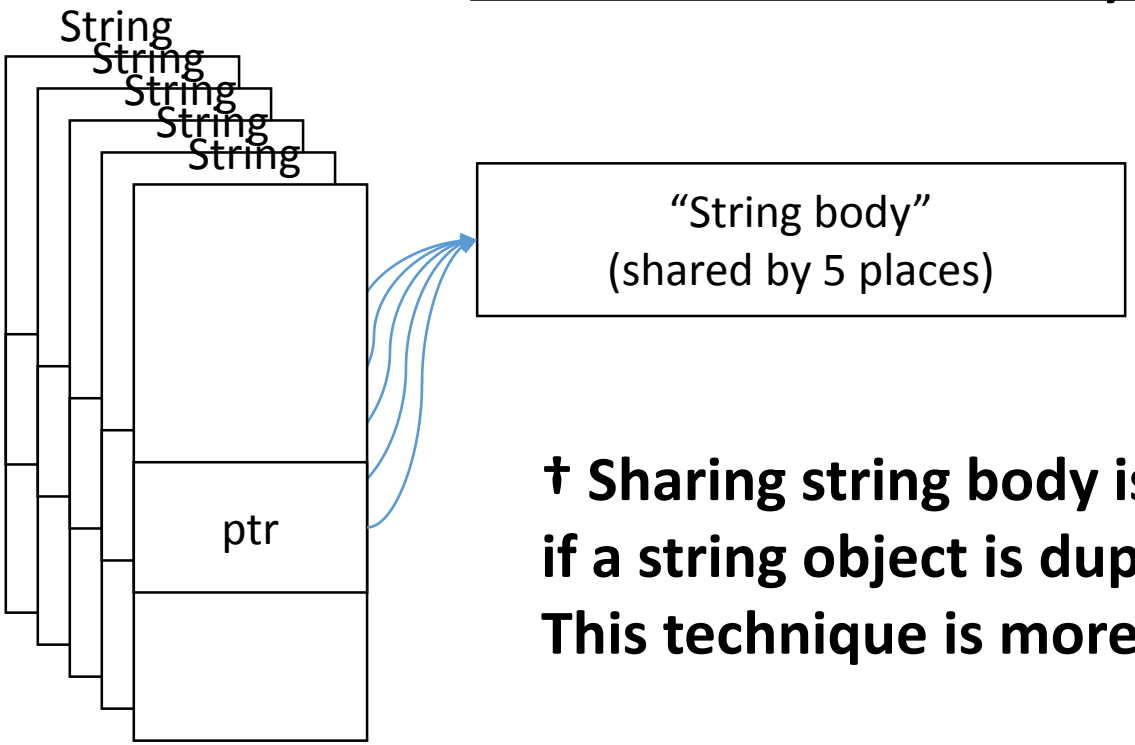
- For some strings have same “string body”, they has own string body each other.



Memory efficient string management

- It can be shared by strings w/ dirty bit.

→ Reduce memory consumption!!

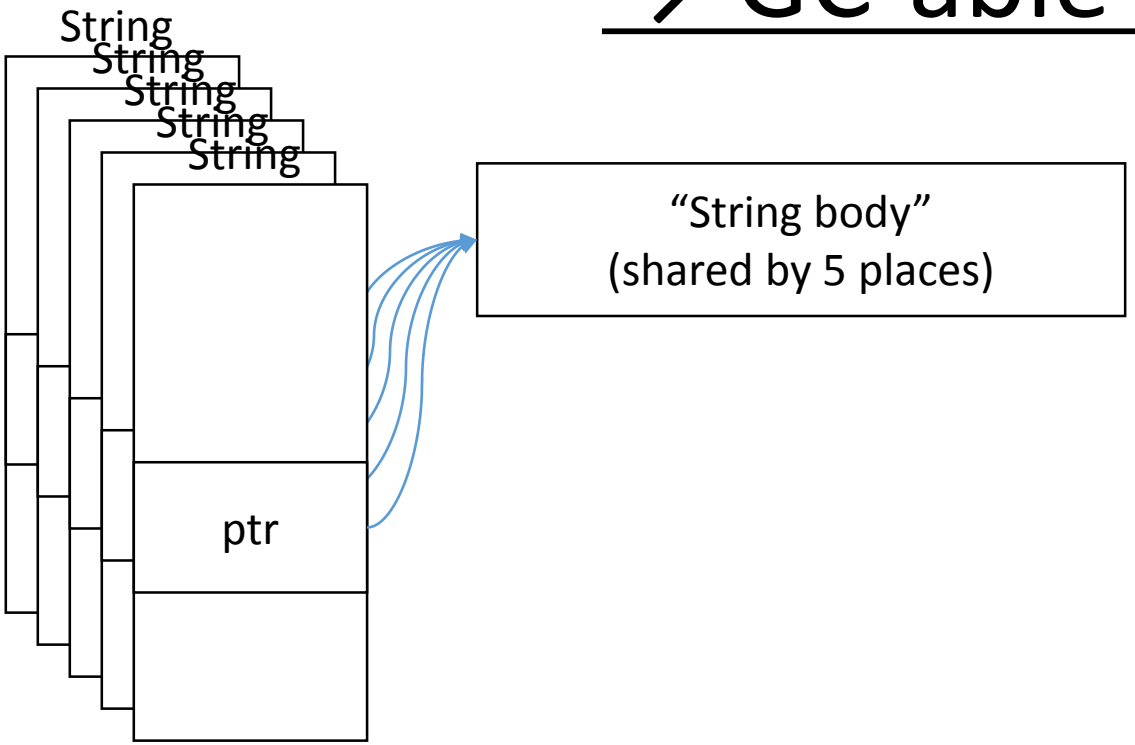


† Sharing string body is implemented now if a string object is duped.
This technique is more aggressive approach.

Memory efficient string management

- This mechanism can work with Symbol management

→ GC-able Symbol



Questions and answers

Questions and Answers RGenGC and CoW friendly

- No problem because only touch flags for oldgen and shady

Questions and Answers
GC + Threads

- Parallel GC
  - Run GC process in parallel (simultaneously)
  - Parallel marking
  - Parallel sweeping (in today's talk)
- Concurrent GC / Incremental GC
  - Run ruby threads (mutator threads) and GC threads concurrently
  - Major GC consumes huge time (same as current GC) → Need concurrent GC to reduce pause time
  - New WB API is also designed for concurrent GC

Agenda

- Ruby's rough history
- Ruby 2.1 new "internal" features
  - Internal object management hooks
    - Object allocation tracing
    - GC hooks
  - **RGenGC: Restricted Generational Garbage Collection ← Today's main topic**
- Ruby 2.1 expected "internal" features
  - Parallel sweeping
  - Sophisticated inline cache invalidation mechanism
  - Memory efficient string management

Summary

- We are implementing new features and improving Ruby's quality for Ruby 2.1
- Especially introducing new "Generational garbage collector" will achieve huge performance improvement
- Ruby 2.1 is currently scheduled on Dec 25, 2013. Don't miss it!

Thank you

Koichi Sasada

Heroku, Inc.

<ko1@heroku.com>