Ruby 2.4 Internals

Koichi Sasada

ko1@cookpad.com

cookpad

Ruby 2.4.0 Released

Posted by naruse on 25 Dec 2016

We are pleased to announce the release of Ruby 2.4.0.

Ruby 2.4.0 is the first stable release of the Ruby 2.4 series. It introduces many new features, for example:

Introduce hash table improvement (by Vladimir Makarov)

Improve the internal structure of hash table (st_table) by introducing open addressing and an inclusion order array. This improvement has been discussed with many people, especially with Yura Sokolov.

Binding#irb: Start a REPL session similar to `binding.pry`

While you are debugging, you may often use `p` to see the value of variables. With pry you can use `binding.pry` in your application to launch a REPL and run any Ruby code. r56624 introduces `binding.irb` which behaves like that with irb.

Note:
Ruby 2.4.0 has several bugs.

Note2:
No x.y.0 doesn't have several bugs.

Ruby 2.4.0

Ruby
A PROGRAMMER'S BEST FRIEND

Google™ Custom Search | Search

Downloads | Documentation | Libraries | Community | **News** | Security | About Ruby

Ruby 2.4.0 Released

ed by se o ec 2

are sed nou the re f Ruby 2.4.0.

Ruby 2.4.0 is the first stable release he Ruby 2.4 series. It introduces many new features, for example:

Introduce hash table improvement (by Vladimir Makarov)

Improve the internal structure of hash table (st_table) by introducing open addressing and an inclusion order array. This improvement has been discussed with many people, especially with Yura Sokolov.

Binding#irb: Start a REPL session similar to `binding.pry`

While you are debugging, you may often use `p` to see the value of variables. With pry you can use `binding.pry` in your application to launch a REPL and run any Ruby code. r56624 introduces `binding.irb` which behaves like that with irb.

ecent News

- uby 2.4.0 Relea
- y 2.4.0-rc1 R ed
- uby 2.3 elease
- Ruby 2.3.2 Released
- Ruby 2.2.6 Released

Syndicate

Recent News (RSS)

New features written in a release announcement

- **Introduce hash table improvement (by Vladimir Makarov)**
- **Binding#irb: Start a REPL session similar to binding.pry**
- **Unify Fixnum and Bignum into Integer**
- **String supports Unicode case mappings**
- **Performance improvements**
  - **Array#max, Array#min**
  - **Regexp#match?**
  - **speed up instance variable access**
- **Debugging**
  - **Thread#report_on_exception and Thread.report_on_exception**
  - **Thread deadlock detection now shows threads with their backtrace and dependency**
- **Other notable changes since 2.3**
  - **Support OpenSSL 1.1.0 (drop support for 0.9.7 or prior)**
  - **ext/tk is now removed from stdlib Feature #8539**
  - **XMLRPC is now removed from stdlib Feature #12160**

Ruby 2.4 Internals

Koichi Sasada

ko1@cookpad.com

Jonan Scheffler

DEVELOPER ADVOCATE

Heroku

Heroku Staff

Ruby 2.4 Released: Faster Hashes, Unified Integers and Better Rounding

December 25, 2016 by Jonan Scheffler

The Ruby maintainers continued their annual tradition by gifting us a new Ruby version to celebrate the holiday: Ruby 2.4 is now available and you can try it out on Heroku.

Ruby 2.4 brings some impressive new features and performance improvements to the table, here are a few of the big ones:

Continue reading »

https://blog.heroku.com/ruby-2-4-features-hashes-integers-rounding

Any other topics?

Aaron Patterson
@tenderlove

フォロー中

People should really upgrade to Ruby 2.4. We're seeing GC time cut in half for the same throughput of allocations

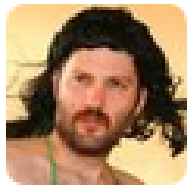
英語から翻訳

223 リツイート　348 いいね

1:11 - 2017年3月10日

19　223　348

!?



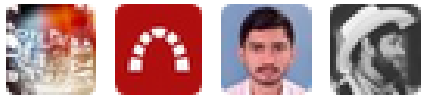
Aaron Patterson
@tenderlove

フォロー中

@schneems likely #10212 in redmine. But could be hash restructuring. Two possibilities. (not sure if I care though! ;))

英語から翻訳

4
いいね



1:33 - 2017年3月10日

Bug #10212

編集 ウォッチ

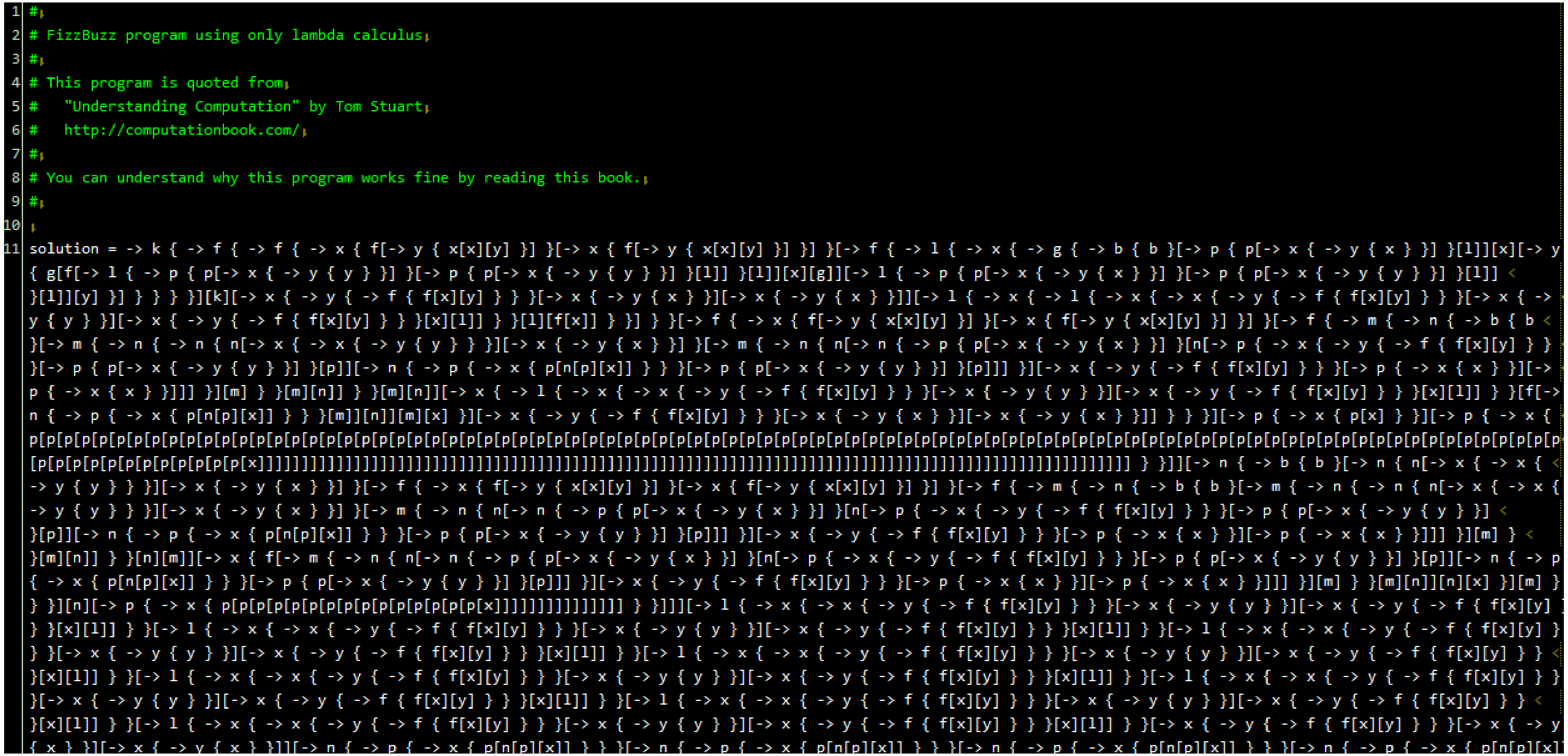
MRI is not for lambda calculus

ko1 (Koichi Sasada) が2年以上前に追加. 約1ヶ月前に更新.

ステータス:	Closed		
優先度:	Normal		
担当者:	ko1 (Koichi Sasada)		
対象バージョン:	-		
ruby -v:	ruby 2.2.0dev (2014-08-21 trunk 47241) [x86_64-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

[ruby-core:64838]

benchmark/bm_app_lc_fizzbuzz.rb



```
#
# FizzBuzz program using only lambda calculus
#
# This program is quoted from
#   "Understanding Computation" by Tom Stuart
#   http://computationbook.com/
#
# You can understand why this program works fine by reading this book.
#

solution = -> k { -> f { -> f { -> x { f[-> y { x[x][y] }] }[-> x { f[-> y { x[x][y] }] }] }[-> f { -> l { -> x { -> g { -> b { b }[-> p { p[-> x { -> y { x } }] }[l]][x][-> y
{ g[f[-> l { -> p { p[-> x { -> y { y } }] }[-> p { p[-> x { -> y { y } }] }[l]] }[l]][x][g]][-> l { -> p { p[-> x { -> y { x } }] }[-> p { p[-> x { -> y { y } }] }[l]]
}[l]][y] }] } } } }][k][-> x { -> y { -> f { f[x][y] } } }[-> x { -> y { x } }][-> x { -> y { x } }]][-> l { -> x { -> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { ->
y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[l][f[x]] } }] } }[-> f { -> x { f[-> y { x[x][y] }] }[-> x { f[-> y { x[x][y] }] }] }[-> f { -> m { -> n { -> b { b
}[-> m { -> n { -> n { n[-> x { -> x { -> y { y } } }][-> x { -> y { x } }] }[-> m { -> n { n[-> n { -> p { p[-> x { -> y { x } }] }[n[-> p { -> x { -> y { -> f { f[x][y] } }
}[-> p { p[-> x { -> y { y } }] }[p]][-> n { -> p { -> x { p[n[p][x]] } } }[-> p { p[-> x { -> y { y } }] }[p]]] }][-> x { -> y { -> f { f[x][y] } } }[-> p { -> x { x } }][->
p { -> x { x } }]]] }][m] } }[m][n]] } }[m][n]][-> x { -> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[f[->
n { -> p { -> x { p[n[p][x]] } } }[m]][n]][m][x] }][-> x { -> y { -> f { f[x][y] } } }[-> x { -> y { x } }][-> x { -> y { x } }]] } } }][-> p { -> x { p[x] } }][-> p { -> x {
p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[p
[p[p[p[p[p[p[p[p[p[p[p[p[x]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]] } }]][-> n { -> b { b }[-> n { n[-> x { -> x {
-> y { y } } }][-> x { -> y { x } }] }[-> f { -> x { f[-> y { x[x][y] }] }[-> x { f[-> y { x[x][y] }] }] }[-> f { -> m { -> n { -> b { b }[-> m { -> n { -> n { n[-> x { -> x {
-> y { y } } }][-> x { -> y { x } }] }[-> m { -> n { n[-> n { -> p { p[-> x { -> y { x } }] }[n[-> p { -> x { -> y { -> f { f[x][y] } } }[-> p { p[-> x { -> y { y } }]
}[p]][-> n { -> p { -> x { p[n[p][x]] } } }[-> p { p[-> x { -> y { y } }] }[p]]] }][-> x { -> y { -> f { f[x][y] } } }[-> p { -> x { x } }][-> p { -> x { x } }]]] }][m] }
}[m][n]] } }[n][m]][-> x { f[-> m { -> n { n[-> n { -> p { p[-> x { -> y { x } }] }[n[-> p { -> x { -> y { -> f { f[x][y] } } }[-> p { p[-> x { -> y { y } }] }[p]][-> n { -> p
{ -> x { p[n[p][x]] } } }[-> p { p[-> x { -> y { y } }] }[p]]] }][-> x { -> y { -> f { f[x][y] } } }[-> p { -> x { x } }][-> p { -> x { x } }]]] }][m] } }[m][n]][n][x] }][m] }
} }][n][-> p { -> x { p[p[p[p[p[p[p[p[p[p[p[p[p[p[p[x]]]]]]]]]]]]]]] } }]]][-> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] }
} }[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] }
} }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } }
}[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] } }
}[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } }
}[x][l]] } }[-> l { -> x { -> x { -> y { -> f { f[x][y] } } }[-> x { -> y { y } }][-> x { -> y { -> f { f[x][y] } } }[x][l]] } }[-> x { -> y { -> f { f[x][y] } } }[-> x { -> y
{ x } }][-> x { -> y { x } }]][-> n { -> p { -> x { p[n[p][x]] } } }[-> n { -> p { -> x { p[n[p][x]] } } }[-> n { -> p { -> x { p[n[p][x]] } } }[-> n { -> p { -> x { p[n[p][x]
```

アンダースタンディング コンピュテーション
——単純な機械から不可能なプログラムまで

Tom Stuart　著、笹田 耕一　監訳、笹井 崇司　訳

2014年09月 発行
336ページ
ISBN978-4-87311-697-6
フォーマット Print PDF
原書: Understanding Computation

オライリー・ジャパンで書籍を購入:
定価3,456円

カートに入れる

Ebook Storeで電子版を購入:
価格2,765円

カートに入れる

Bug #10212 MRI is not for lambda calculus

```
jruby 1.7.12 (1.9.3p392) 2014-04-15 643e292 on OpenJDK 64-Bit Server VM 1.7.
real    0m26.648s
user    0m30.091s
sys     0m4.369s

mruby 89e9df26819b9555fb790a16662f4ad2b9cbb2e2
real    0m27.145s
user    0m27.110s
sys     0m0.012s

ruby 2.2.0dev (2014-08-21 trunk 47241) [x86_64-linux]
real    1m54.648s
user    1m54.512s
sys     0m0.028s
```

JRuby 26 sec

mruby 27 sec

MRI 114 sec

Feature #12628
change block/env structs

Feature #12628

change block/env structs

« 前 | 2/122 | 次 »

ko1 (Koichi Sasada) が8ヶ月前に追加. 8ヶ月前に更新.

ステータス:	Closed
優先度:	Normal
担当者:	ko1 (Koichi Sasada)
対象バージョン:	-

[ruby-core:76568]

Ruby 2.4 Internals
Change block/env structs

Koichi Sasada

ko1@cookpad.com

Issues

1. we need to clear rb_control_frame_t::block_iseq for every frame setup. It consumes space (a VALUE for each frame) and initializing time.
2. There are several block passing ways by ISeq (iter{...}), Proc(iter(&pr)), Symbol(iter(:sym)). However, they are not optimized (for Symbol blocks, there is only ad-hoc check code).
3. Env (and Proc, Binding) objects are not WB-protected ([Bug #10212]).

Method dispatch (etc) improvements

Cleanup src code

<u>Improve GC perf.</u>

Patch

- https://github.com/ruby/ruby/compare/trunk...ko1:block_code
- “Showing with **1,863 additions** and **1,070 deletions**.”

Approaches

- For (1), (2)
  - Introduce Block Handler (BH)
  - Using BH
- For (3)
  - <u>Introduce Write Barriers (WB) for Env objects</u>

WB protected or unprotected?

- Ruby 2.1.0 introduced Generational GC
  - Only newer objects
  - GenGC requires “Write barriers” (WB), but MRI allows WB unprotected objects

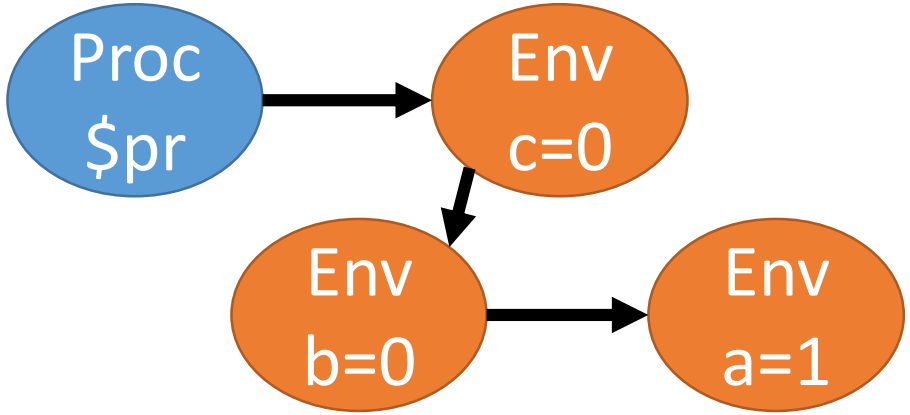
    (See my past presentations for details)

  - WB protected objects: GenGC → Fast
  - WB unprotected objects: Not GenGC → Slow

RubyVM::Env objects

- Env objects represent captured local variables
  - Each Proc or Binding has <u>at least</u> one Env object
  - Proc object "$pr" consists of 3 Env objects

```
a = 1
1.times{|b|
  1.times{|c|
    $pr = Proc.new{
      # you can access a, b, c
    }
  }
}
```



RubyVM::Env objects were WB-unprotected

- They were WB unprotected because:
  - Difficulty of implementation
  - Performance issue

Performance issue
Assignment performance

- Ruby 2.3 or before

```
*(ep - idx) = val;
```

- Naïve implementation

```
#define VM_EP_IN_HEAP_P(th, ep) ¥
    (!((th)->stack <= (ep) && ¥
       (ep) < ((th)->stack + (th)->stack_size)))

if (VM_EP_IN_HEAP_P(ep)) {
   RB_OBJ_WRITE(VM_ENV_EP_ENVVAL(ep),
                        ep-idx, val);
}
else *(ep - idx) = val;
```

Ideas

1. Lightweight escape detection
2. Skip WB except really required timing

Idea
Lightweight escape detection

- **Move** `cfp->flags` **to** `ep[0]`
- Introduce a VM_ENV_FLAG_ESCAPED flag to represent escaped Env.

- `// Before`

```
#define VM_EP_IN_HEAP_P(th, ep)   (!((th)->stack <= (ep) && (ep) < ((th)->stack + (th)->stack_size)))
```

- `// After`

```
#define VM_EP_IN_HEAP_P(ep)      (ep[0] & VM_ENV_FLAG_ESCAPED)
```

Idea
Skip WB except really required timing

1. At initializing Env objects, VM_ENV_FLAG_WB_REQUIRED is true.
2. At first local variable assignment, VM_ENV_FLAG_WB_REQUIRED is true, we remember this Env object forcibly. **And turn off this flag.**
3. At next local variable assignment, VM_ENV_FLAG_WB_REQUIRED is false, so we can ignore WB protection.
4. **At GC marking for this Env object, we turn on** VM_ENV_FLAG_WB_REQUIRED and goto (2).

Very danger technique because it depends on GC implementation

Naïve code

```
#define VM_EP_IN_HEAP_P(th, ep)    (!((th)->stack <= (ep)
&& (ep) < ((th)->stack + (th)->stack_size)))

vm_env_write(const VALUE *ep, int index, VALUE v) {
  if (VM_EP_IN_HEAP_P(ep)) {
    RB_OBJ_WRITE(VM_ENV_EP_ENVVAL(ep), ep-idx, val);
  }
  else {
    *(ep - idx) = val;
  }
}
```

Final code

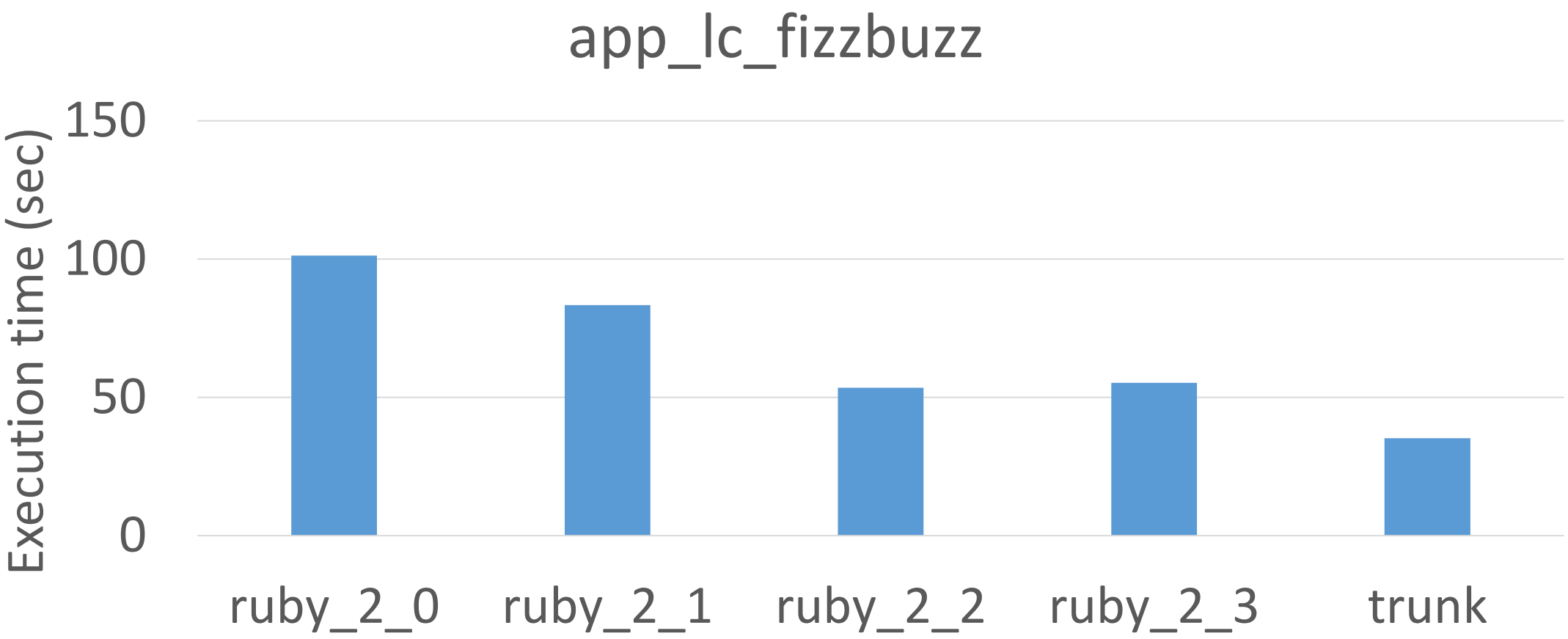
```
vm_env_write(const VALUE *ep, int index, VALUE v) {
  VALUE flags = ep[VM_ENV_DATA_INDEX_FLAGS];
  if (LIKELY((flags & VM_ENV_FLAG_WB_REQUIRED) == 0)) {
    *(ep - idx) = val; /* mostly used */
  }
  else {
    /* remember env value forcibly */
    vm_env_write_slowpath(ep, index, v);
  }
}
```

Benchmark result

```
                    trunk  modified
app_lc_fizzbuzz    58.277    41.729 (sec) (x 1.397 faster)
vm1_simplereturn*   0.660     0.638 (sec) (x 1.035 faster)
vm1_yield*          0.738     0.650 (sec) (x 1.135 faster)
```

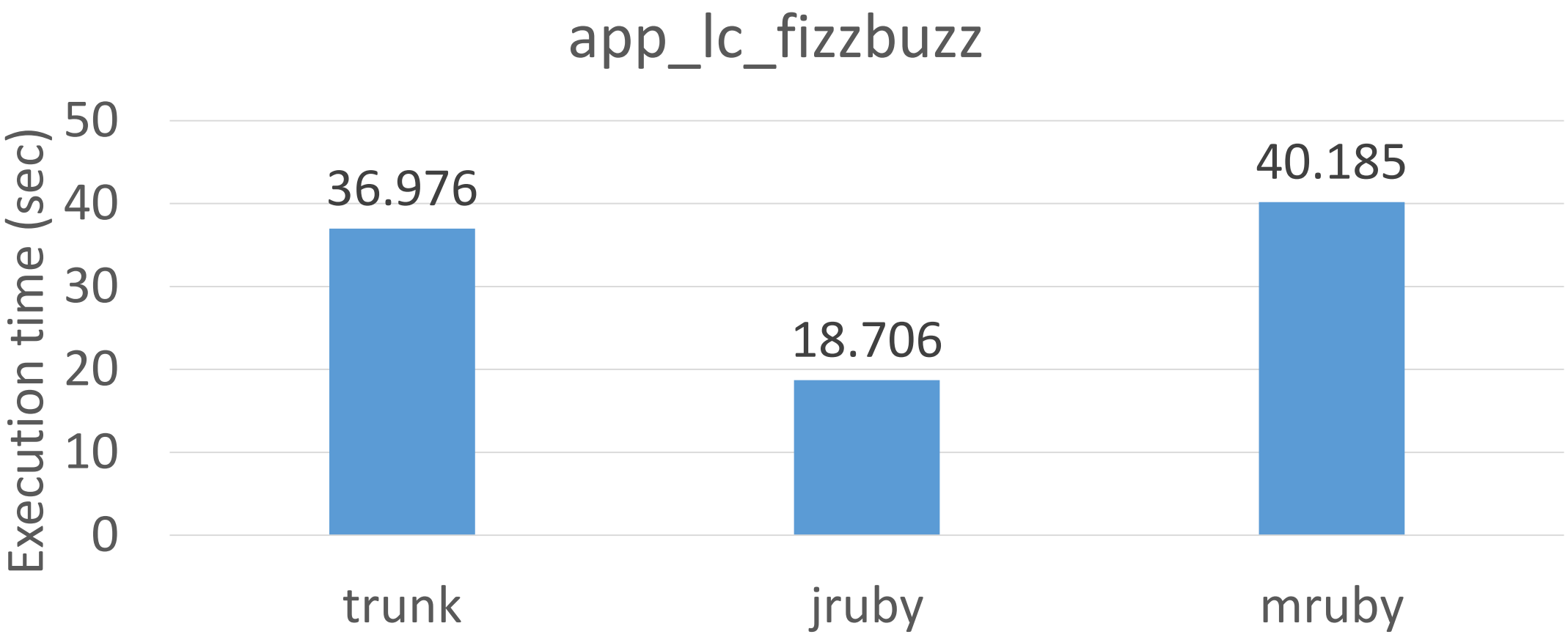
Bug #10212 MRI is not for lambda calculus

lc_fizzbuzz with MRI versions



Bug #10212 MRI is not for lambda calculus

lc_fizzbuzz with MRI, JRuby, mruby



Summary

- Ruby 2.4.0 has many improvements
- Now Proc (Env) objects are WB protected and we have more faster GC (marking)
- My ideas allow to protect Env objects without big performance impact

Thank you for your attention

Koichi Sasada

<ko1@cookpad.com>

cookpad