Rubyによる本気のGC

Serious GC with Ruby

@nari3

#sprk2012

2012/9/15

ネットワーク応用通信研究所

提供

うなぎの件

とにかくプログラミング(や鰻)が好きな人ウォンテッド

株式会社ネットワーク応用通信研究所

埋め込む Tweet Like 0 Send SHARE

<URL:https://www.wantedly.com/projects/478>

自己紹介

- ✓ 中村成洋/@nari3/nari/authorNari
- ✓ CRubyのコミッタ
- ✓ GCメンテナ＆デストロイヤ

GCとは（ｒｙ

GoogleTrendでみるGC

Google トレンド

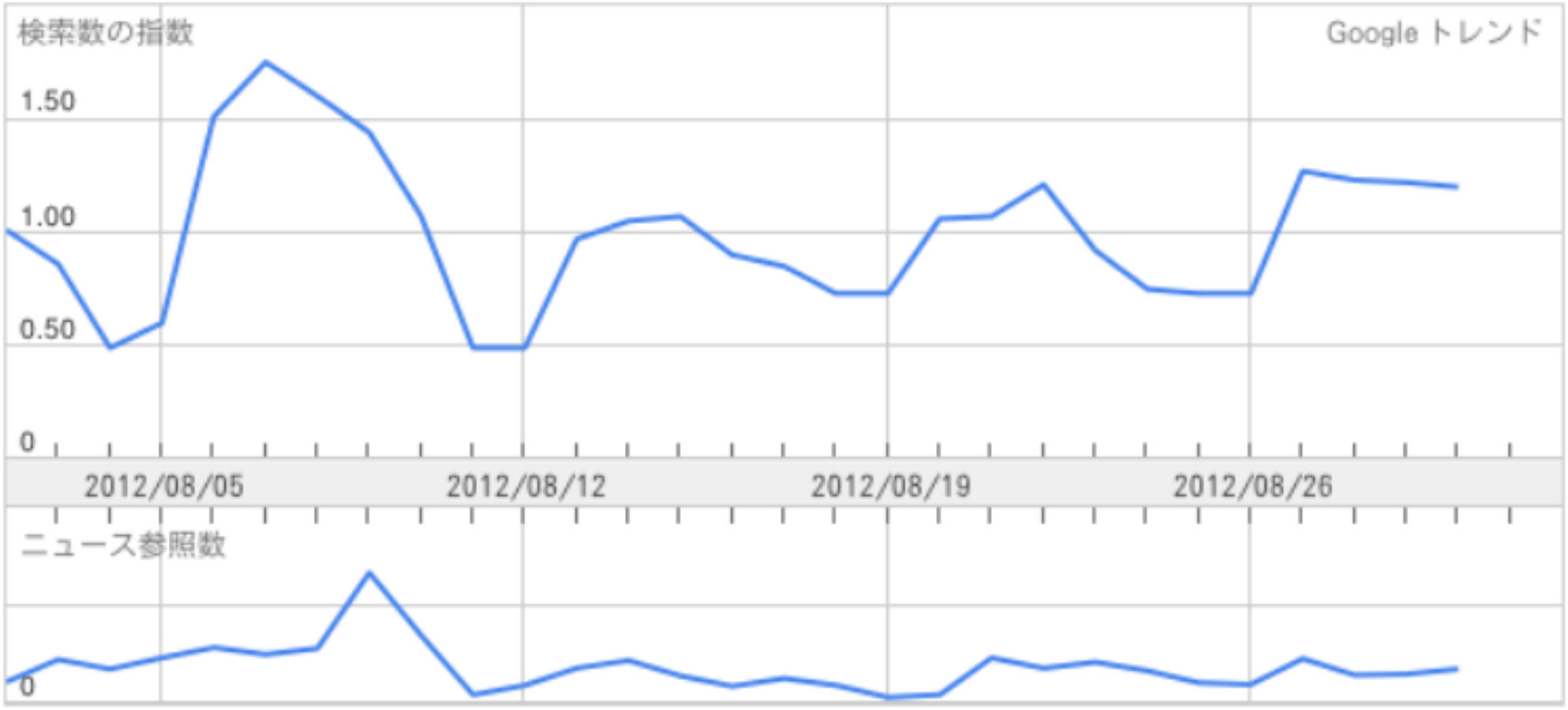
garbage collection

トレンドを検索

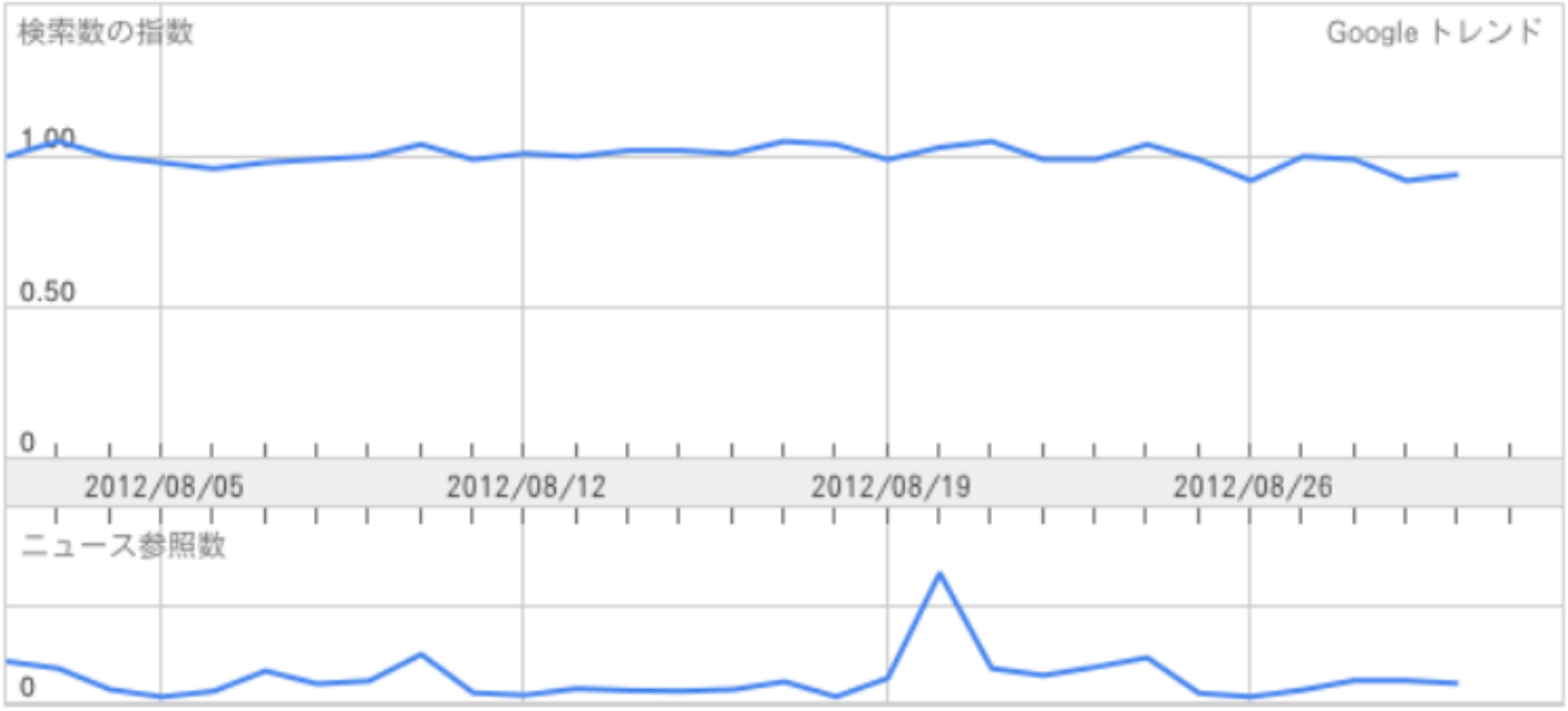
複数のキーワードを比較する場合はカンマか読点で区切ってください。

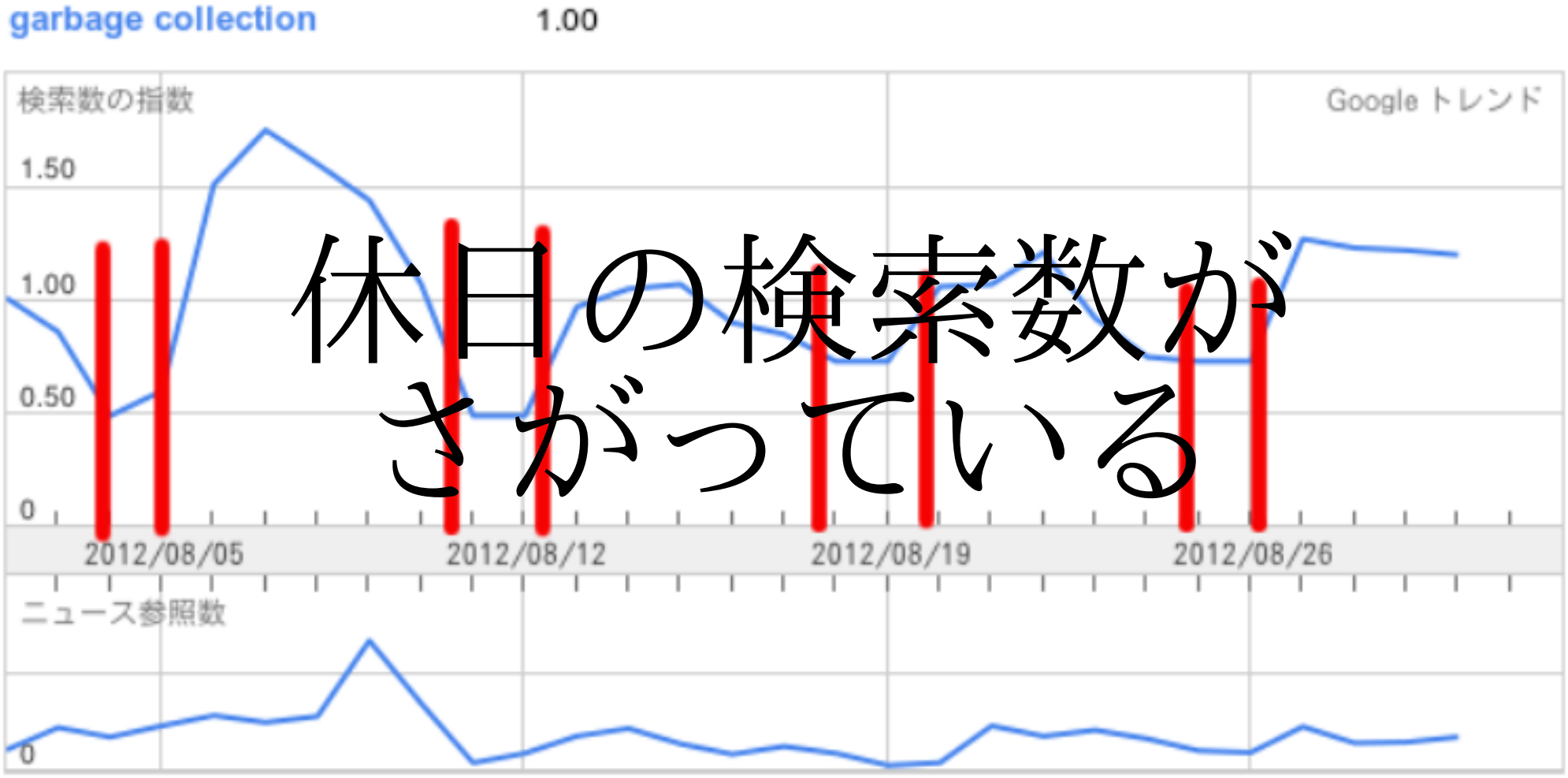
トレンドの推移

garbage collection 1.00



ruby 1.00


検索数の指数
Google トレンド
1.00
0.50
0
2012/08/05
2012/08/12
2012/08/19
2012/08/26
ニュース参照数
0



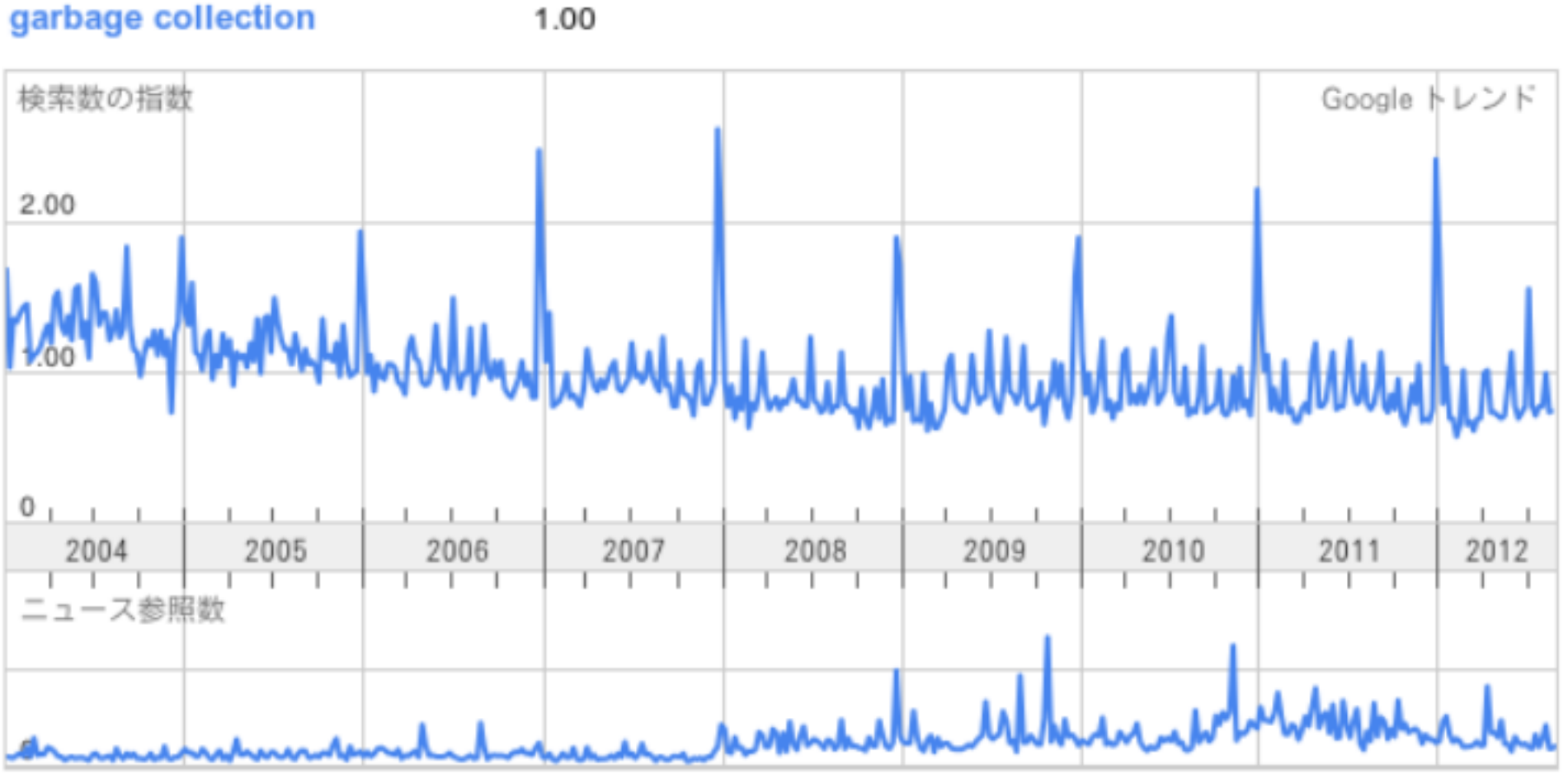
休日の検索数がさがっている

つまり

- ✓ 平日
  - ✓ お仕事でGCを作るリア充
  - ✓ もしくはGCバグに悩まされる一般ピープルが多い
- ✓ 休日
  - ✓ 趣味でGCをいじる人が少ない

GCはまだまだ愛されていない

ちなみに:年末も異常に検索されている



リアルGCの恐れ..

とにかく
GCをもっと
よく知りましょう！

徹底解剖G1GC
実装編
正式版公開！！
（無料配布）

徹底解剖
「G1GC」
実装編

この場を借りて本書のスポンサーのみなさまありがとうございます

m(_ _)m

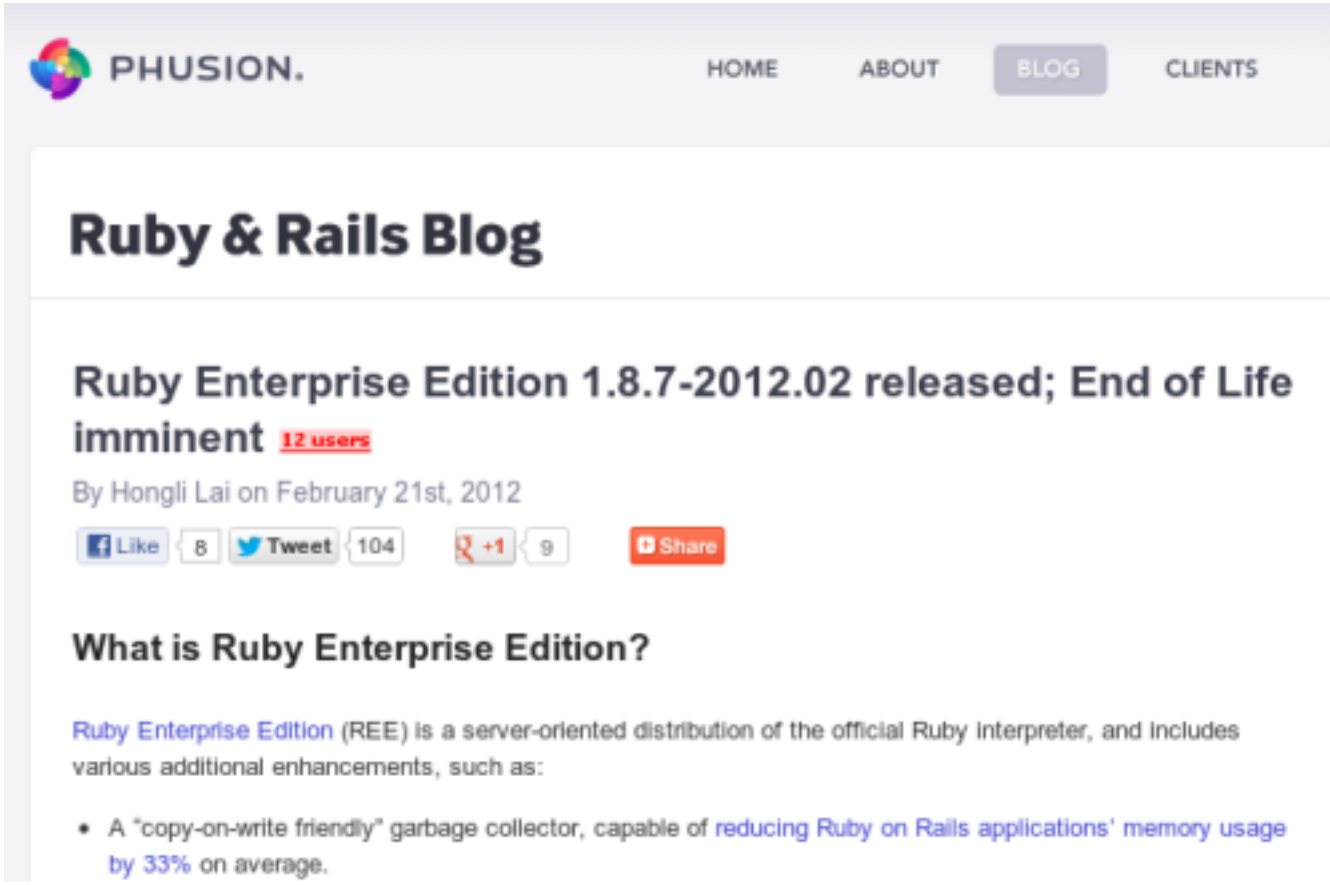
Ruby2.0 の GC update

偉い人がいってた

http://www.flickr.com/photos/recompile_net/5952564080

僕もなにか潰したい！

REE is dead



<URL:http://blog.phusion.nl/2012/02/21/ruby-enterprise-edition-1-8-7-2012-02-released-end-of-life-imminent/>

End of Life

Support for Ruby 1.8.x is slowly being dropped by the upstream Ruby core developers in favor of Ruby 1.9 and beyond. The Rails team has recently announced that they will be dropping Ruby 1.8 support in Rails 4. As such, we are also slowly End-of-Life'ing Ruby Enterprise Edition.

We have no plans to create a Ruby 1.9-based version of Ruby Enterprise Edition for the following reasons:

- A copy-on-write patch has recently been checked into Ruby 2.0.

- A copy-on-write patch has recently been checked into Ruby 2.0.

それをいれたのオレオレ

BitmapMarking

- ✓ fork使うようなプログラムで嬉しいはず
- ✓ Passengerなどで恩恵があるらしい

REEつぶしたった
(^o^)

最近の悩み:
Kijiはどうやって潰そうか...

本題

Rubyで**GC**を書くというお話

動機(1)

RubyKaigi Driven Development

過去の**RubyKaigi**の発表...

- ✓ LazySweepGC - RubyKaigi2008
- ✓ LonglifeGC - 2009
- ✓ LazySweepGC - 2010
- ✓ ParallelMarkingGC - 2011

こ、これ全部Cの話じゃないか！

Rubyの話、させてください

動機(2)

卜部昌平のあまりreblogしないtumblr

どうも周知徹底が不足しているようなので再度のお願いとなりますが、C死ね。

- 確かにCでしか書けない類のプログラムは存在する(例を挙げるならKernel)が、それはCの存在を赦す理由にはならない。
- 確かにCに輪をかけてさらにダメな類のプログラミング言語は存在する(例を挙げるならC++)が、それはCの存在を赦す理由にはならない。
- 確かにCでしか書けないダメプログラマは存在する(例を挙げてほしければここにおまえの名前を入れろ)が、それはCの存在を赦す理由にはならない。

結論:C死ね。

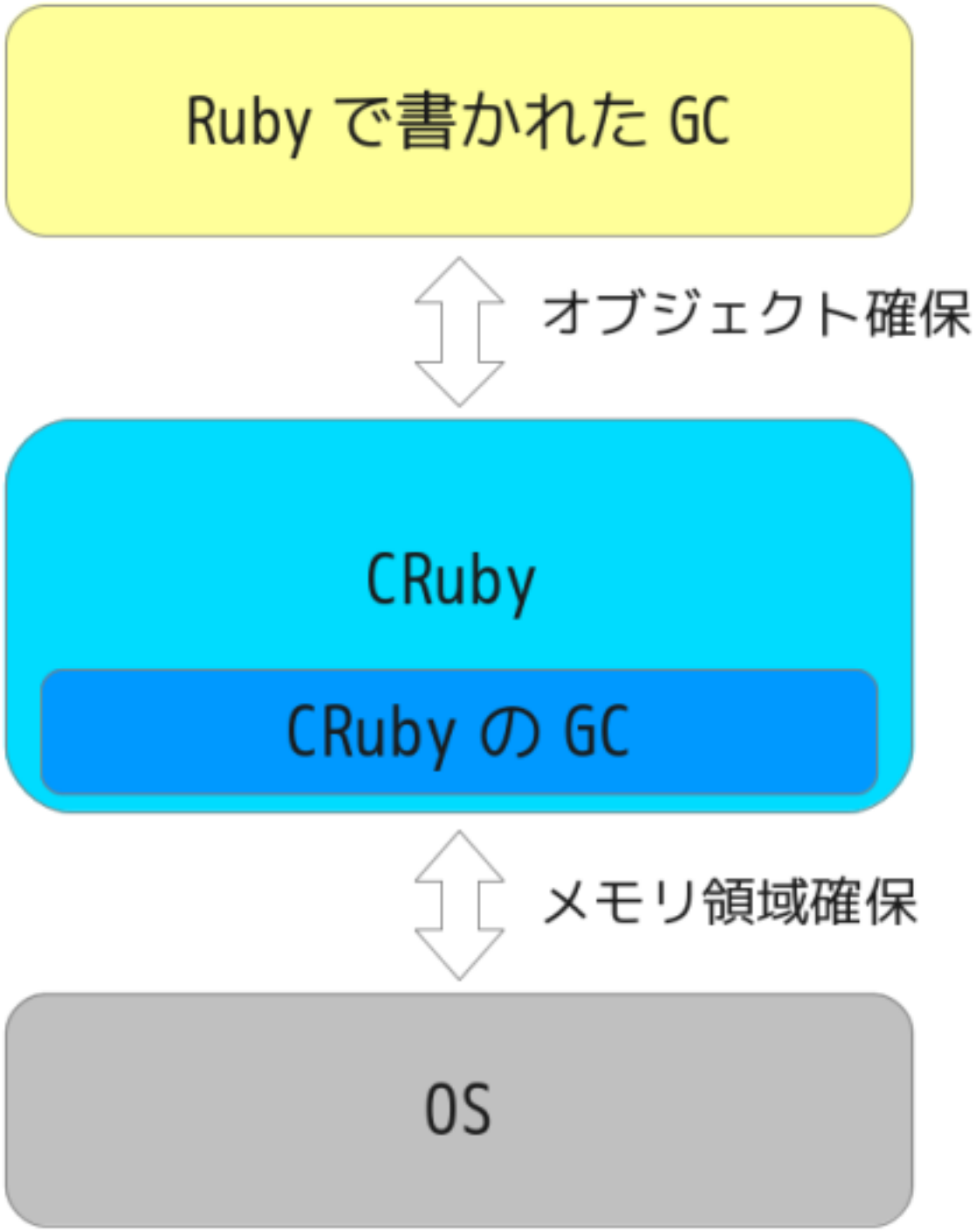
> 結論:C死ね。

[cited from `卜部昌平のあまりreblogしないtumblr']

CRubyのGCはCで書いてる

Rubyで**GC**は書けるんだろうか？

素直に考えると...


Ruby で書かれた GC
オブジェクト確保
CRuby
CRuby の GC
メモリ領域確保
OS

これはあまり意味がない

- 下にあるGCの性能に影響をうける
  - 性能評価しづらい
  - 下にあるGCが遅いと、上のGCも遅くなる

これはあまり意味がない

- 言語処理系にGCは1つあればいいですよ感
  - 無駄に話が複雑になっているような気がする

もっと違うアプローチを
考えよう！

Meta-circular evaluator

An evaluator that is written in the same language that it evaluates is said to be ***metacircular***.

[cited from `4.1 The Metacircular Evaluator - SICP']

簡単な例:
Lispで**Lisp**書いたった

他の具体例を調べてみる

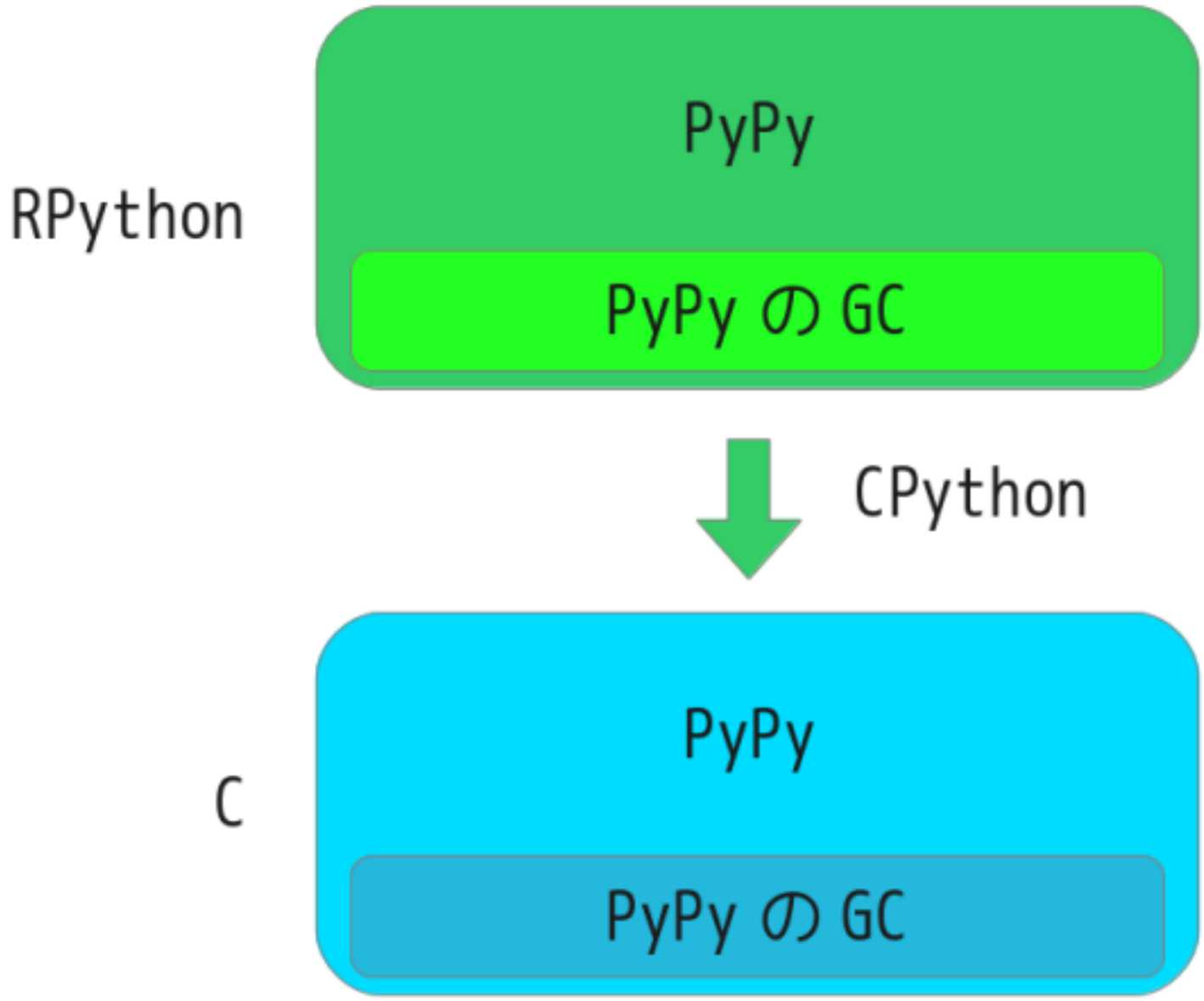

pypy

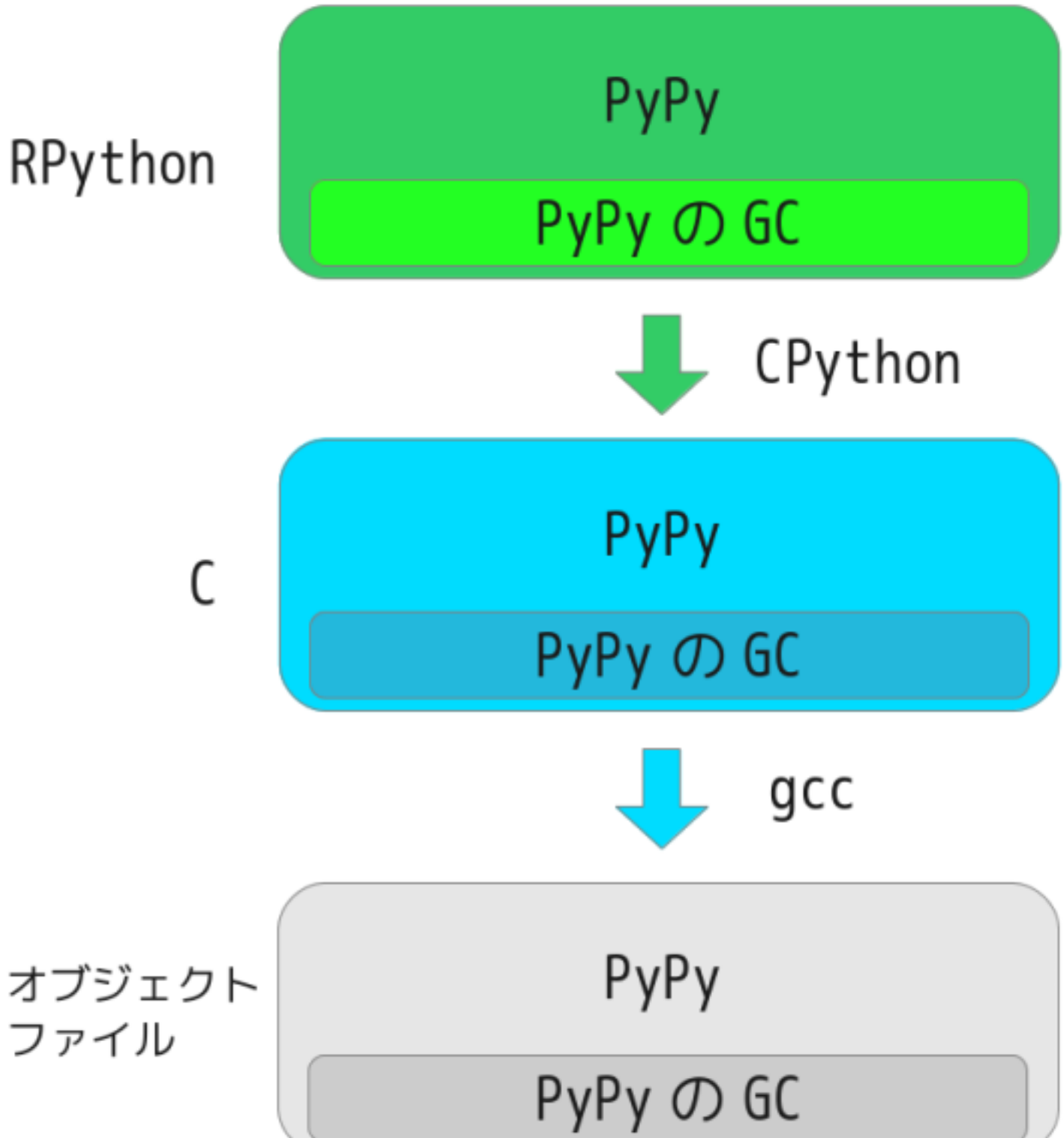
もう少し詳しく

- ✓ RPython(Restricted Python)で実装
  - ✓ Pythonの言語サブセット
- ✓ RPython -> C/LLVM/Java..

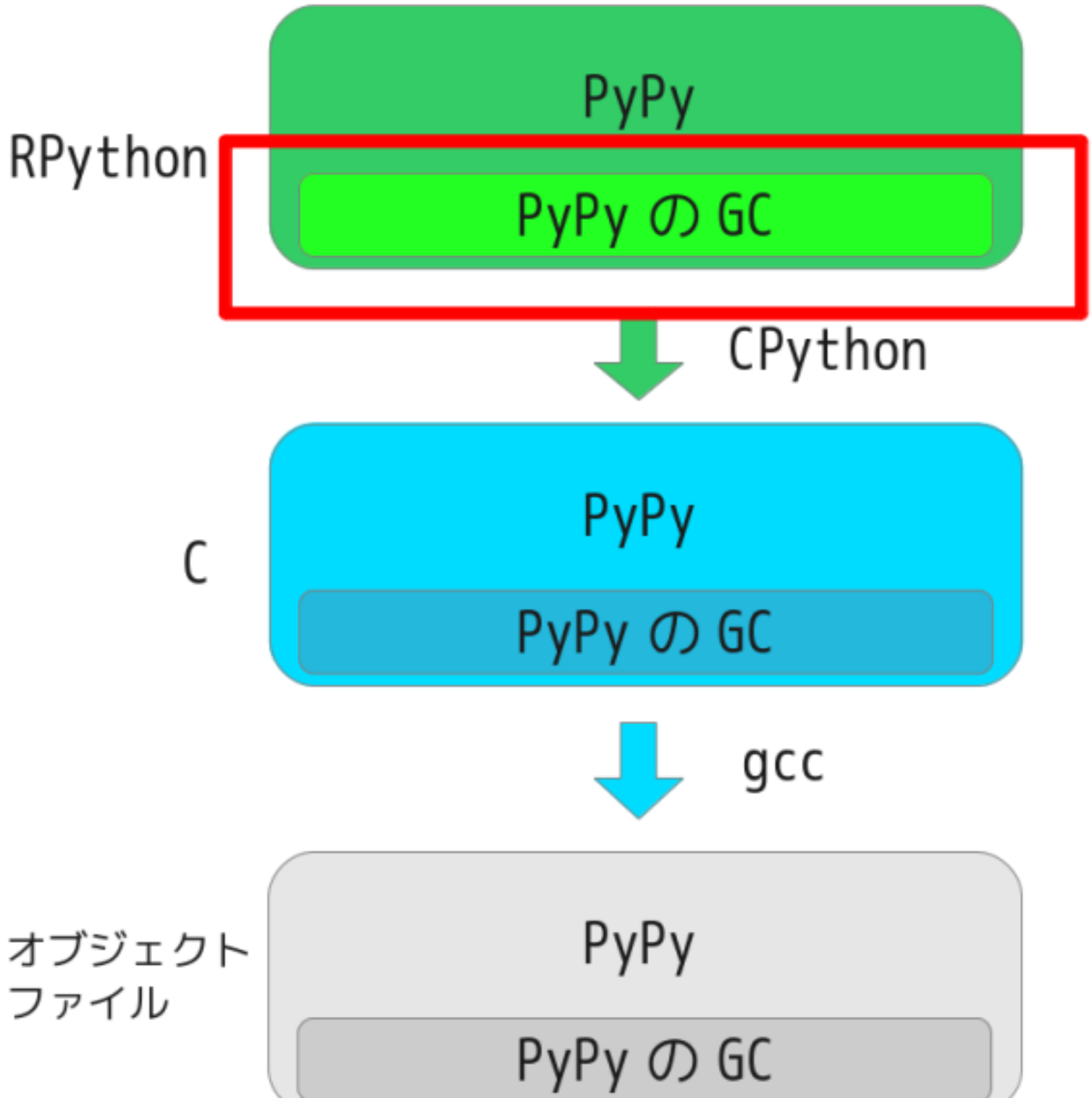
変換イメージ

RPython


RPython
PyPy
PyPy の GC
CPython
C
PyPy
PyPy の GC


RPython
PyPy
PyPy の GC
CPython
C
PyPy
PyPy の GC
gcc
オブジェクト
ファイル
PyPy
PyPy の GC

重要なポイント


RPython
PyPy
PyPy の GC
CPython
C
PyPy
PyPy の GC
gcc
オブジェクト
ファイル
PyPy
PyPy の GC

GCも RPythonで書かれている

疑問

GC動作中のGCはどうするのか？

- GC動作中にGCが起きて、どの動作中にGCが起きて…
  - GC無限ループ...?

RPython

- GC中はGCオブジェクトを作らないように気を付ける（たぶん）
  - mallocなどで直接メモリを切り出す
    - libffiを利用

GC中に**GC**が走ることはない


Rubinius
Use Ruby™

仕組み（ｒｙ

ただし

- GCをRubyで書く仕組みがない
  - 今考えればRubiniusをいじったほうがよかったかも…？
    - まぁいいか...


Jikes RVM

JikesRVM

- RVM(Research Virtual Machine) = 研究用VM
  - GC,VM周りでたくさん論文が書かれている
- GCもJavaで書かれている
- プロジェクト委員の一人がRechard Jones（RJGC著者）

直接オブジェクトファイルを吐く

TODO 図

BootImage+BootImageRunner（コア部）

TODO 図Javaバイトコードを解釈、変換されたJikesRVMで実行

GCはどこに消えた？

- ✓ コア部に一緒に変換される
- ✓ GC中はGCが発生しない
  - ✓ PyPyと同じ理由

JikesRVMの功績はこれだけではない

GC部分は**MMTk**という別部品に切り離されている

MMTk(Memory Management Tool kit)

- ✓ GCを11個保有
- ✓ VMからメモリ管理を別の部品として切り離せる

イメージ図

TODO 図VMの間にMMTkがあってメモリを切り出すような図VMはルートやGC対象のオブジェクトの定義を与える

さらにGCも取り替え可能

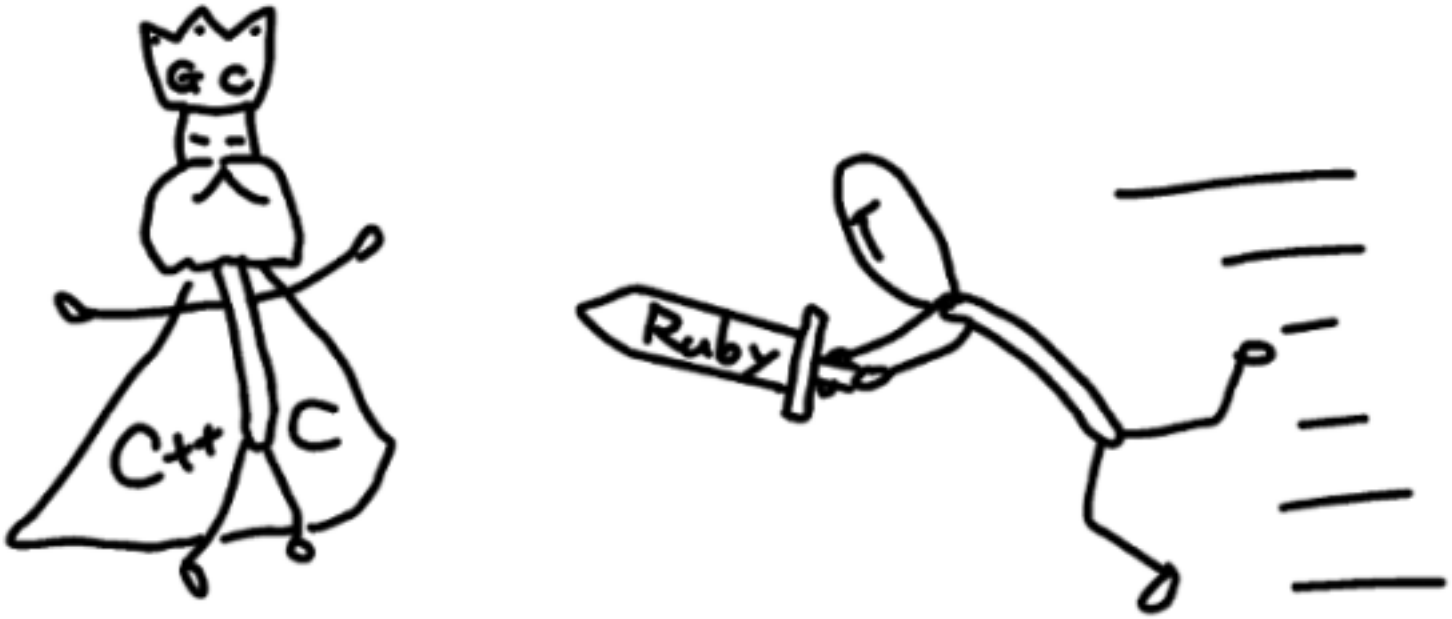
TODO 図上記図に加えてMMTk内部にGCがあるような図

Javaに**Ruby**くっつけられるようなのないかなぁ...

それ、**JRuby**できるよ！

Regicide

Regicde = 国王殺し



MMTkとJRubyを組み合わせるには？

間に**JRuby**を挟めばよい

TODO 図上記図にJRubyを挟んでRubyを使うような図RubyでVM・GCが書ける！

この部分を**Regicde**で提供

TODO 図

この部分はユーザが拡張できる

TODO 図

サンプルコード

オブジェクトの定義

```
class FixnumValue < Regicide::Mutator::ObjectValue
  # ...
end
```

ObjectValueを継承

オブジェクト生成

```
class FixnumValue < Regicide::Mutator::ObjectValue
  def self.from_i(mutator, i)
    # メモリ割り当て
    v = self.new(mutator.alloc(0, 1, mutator.current_stack.pc))
    # Fixnumを格納
    mutator.store_data_field(v.object_value, 0,
      org.vmmagic.unboxed.Word.from_long(i))
    return v
  end
end
```

TODO: メモリ割り当ての図

M&S GCは4行で書ける

```
class MSConstraints < org.mmtk.plan.marksweep.MSConstraints; end
class MS < org.mmtk.plan.marksweep.MS; end
class MSCollector < org.mmtk.plan.marksweep.MSCollector; end
class MSMutator < org.mmtk.plan.marksweep.MSMutator; end
```

動作するサンプル
URL:TODO

苦労話

Java+JRuby

- JRuby側からJavaを使うのは簡単
  - メソッド呼び出し
  - 継承とかとか

Java+JRuby

- ✓ Java側からJRubyを使うのは難しい
  - ✓ けっこう意外だった

どういうことか？

図: JRubyでJavaのメソッドを呼ぶ例

図: 普通に呼び出し

図: JavaからJRubyでメソッドを呼びたい例

どうするの
（？＿？）

RedBridge

RedBridge

» JRuby Project Wiki Home Page

Embedding JRuby

Using Java f[illegible] is JRuby's best-known feature--but you can also go in the other di

[illegible]

embed JRuby in your Java project.

Table of Contents

- JRuby Embed (originally known as Red Bridge)
  - Features of Red Bridge
    - Context Type

RedBridge

- @yokoletさん作（感謝！）
- Javaの中でJRubyの実行環境（コンテナ）を作る
  - コンテナにソースコードぶちこんだり、Rubyコード片を評価できる

イメージ

図: JavaからJRubyでメソッドを呼びたい例

Regicideでは
この辺の技術を
なんやかんや使ってます

Regicideの悪いところ

VM作るのめんどい

けっきょくVMは書かないといけない

- ✓ GCを評価するためにはある程度ちゃんとしたものが必要
  - ✓ でもVMとか興味ないしさぁ...
- ✓ LISPすら作るのが億劫
- ✓ 飽きてきた

JVMのGCの影響を受ける

JikesRVM TODO:図

JRuby+MMTk TODO:図

JVMのGCに引っ張られる

- ✓ 性能が引っ張られる
  - ✓ GC停止も不可能だしきちんと性能評価しづらそう
    - ✓ 論文とか書きづらそう

別のアプローチしとして

- JikesRVMのブートストラップ部分にJRubyを突っ込む
  - うまく動かなかった
    - JRubyがJikesRVMの秘孔を付いているらしい
    - JikesRVMがまだ未熟

Regicideの良いところ

VMが自分で書ける

言語処理系が自前で書ける

- ✓ GCにすごく優しい処理系とか書ける
- ✓ 言語処理系ひっくるめて検討できる

おいしいところは全部**Ruby**で書ける

- ✓ VM, GC, 全部Rubyで書ける
- ✓ Cみたいに阿鼻叫喚しなくていい（かもしれない）
- ✓ 生産性が100倍（らしい）

勝手に処理系が速くなる感

- ✓ JVM速くなる＝Regicide速くなる
  - ✓ WIN-WIN（笑）
    - ✓ Cでも一緒か…

githubにあげています

<URL:http://github.com/authorNari/regicide>

今後の展開

作ってみて気づいたこと

- GC部分は外っ面がRubyなだけでCとあんまし変わらない
  - これならCで書いても一緒なのでは感

作ってみて気づいたこと

- GC実装用のDSLが欲しいだけなのでは…？
  - 今度はそっち方面で攻めてみたい

まとめ

まとめ

- ✓ やっぱりRubyの話できなかった
- ✓ C死ねと思って作ったRegicideがすでに死にそう

ご清聴ありがとうございました

Q&A