

Rails パフォーマンス 基本のキ

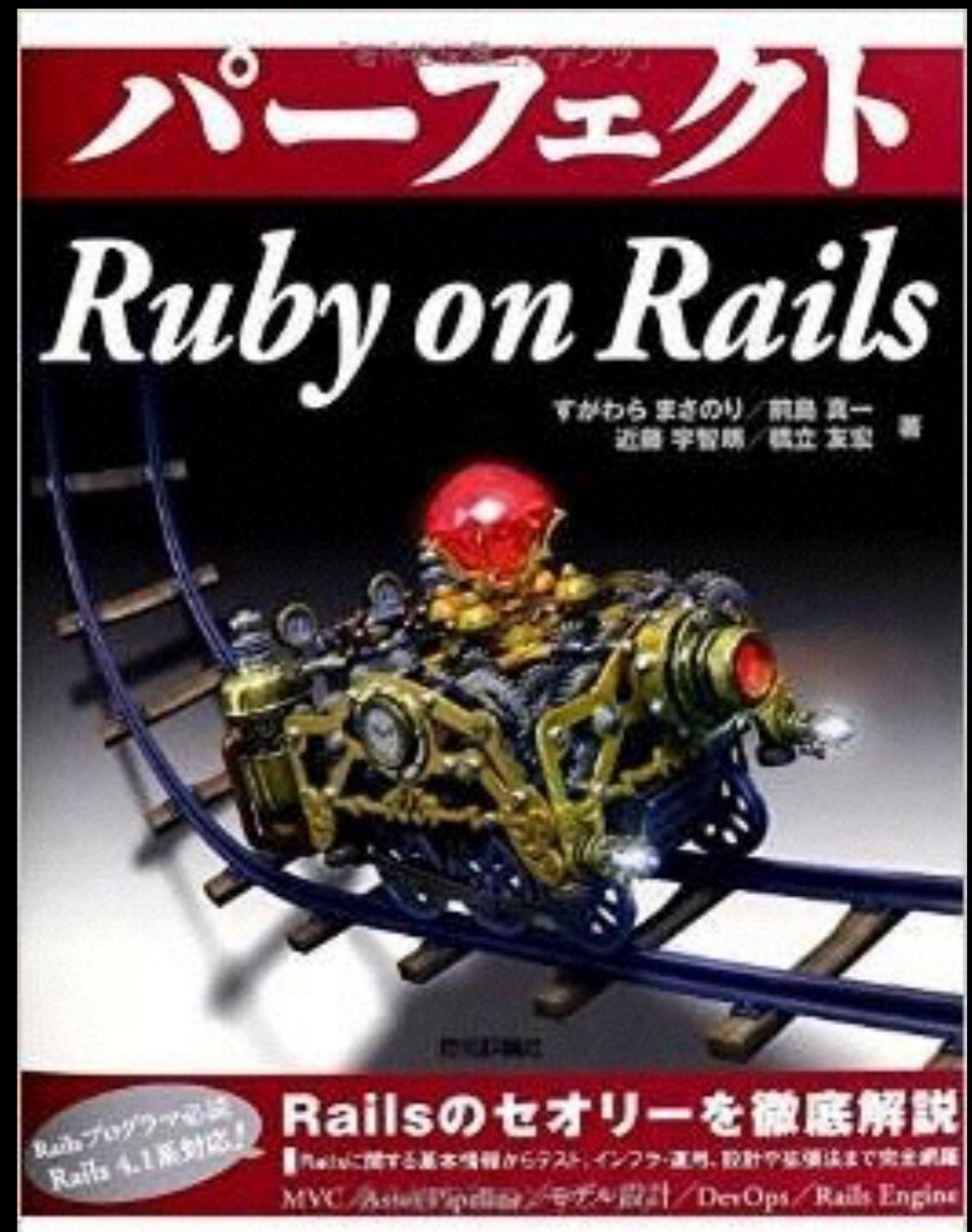
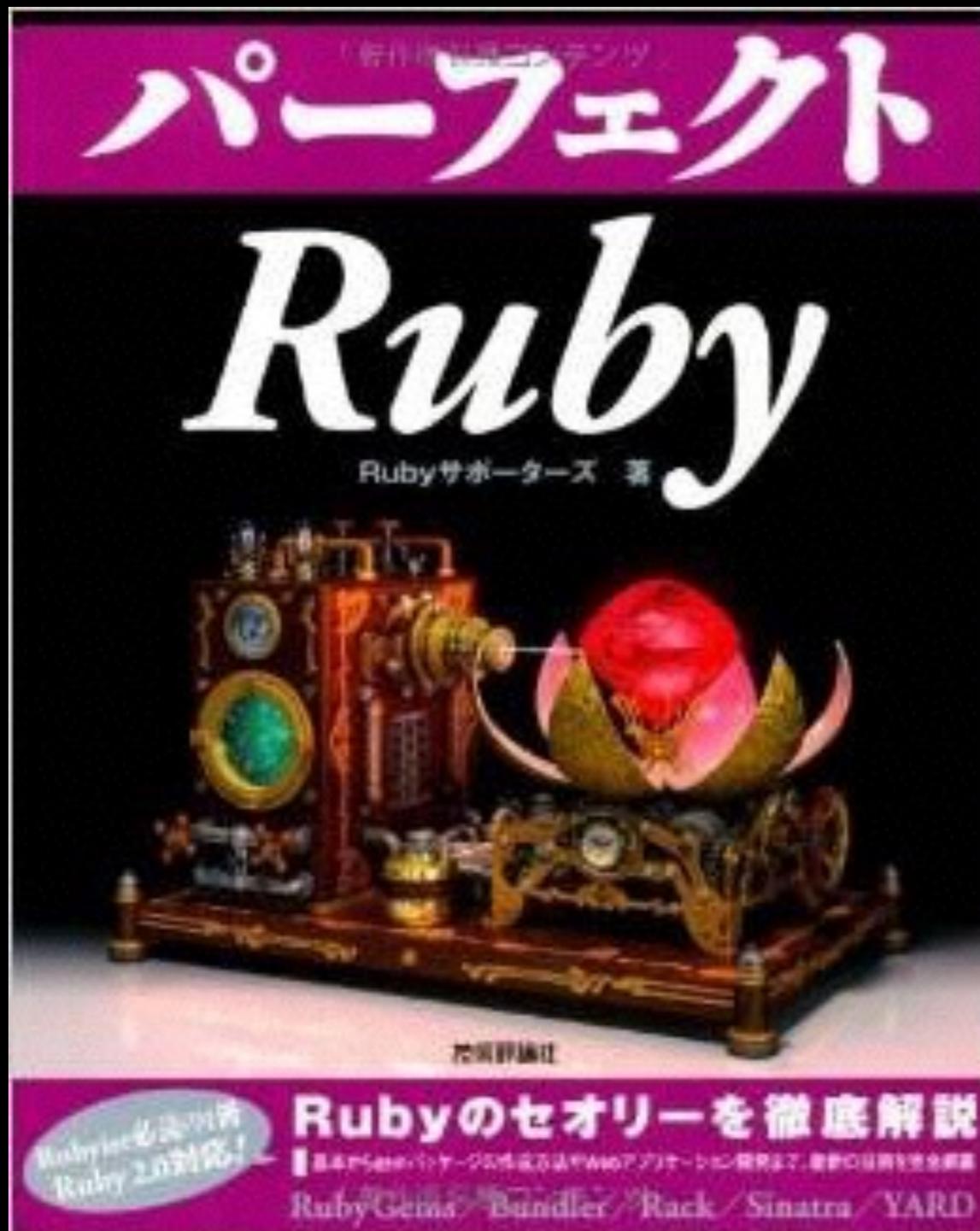
Tomohiro Hashidate
(@joker1007)

p self

=> @joker1007



パRuby パRails



**at Crowdworks
as freelance**

**Crowd
Works**

株式会社クラウドワークス

21世紀の新しいワークスタイルを提供する日本最大級のクラウドソーシング「クラウドワークス」のエンジニアチームです！

161 投稿

6017 ストック
された回数

この内5100が俺

最近の仕事

- Redshiftをバックエンドにした分析基盤
 - esminc/adhoq を弄ってプルリクを出す
- Railsアプリのリファクタリング/改善

**あるページの
表示までの時間**

7sec over

(development)

これはひどい

他人事ではない

A large tortoise, likely a Galapagos tortoise, is walking on a dark, paved surface. The tortoise's shell is dark brown with distinct yellowish-brown spots. The text is overlaid on the shell in a large, white, sans-serif font.

**Railsは遅くないが
何も考えてないと
「遅くなる」**

遅くなる要因

- クソクエリ
- 無駄なクエリ発行とN+1
- ActiveRecordのインスタンス化コスト
- 多過ぎるpartial view
- そもそもビジネスロジックの実装が(ry

例を見ていく

```
2 <b><span class="emphasis"><%= (User.employer_count / 10000.0).round %>  
3
```

クソクエリ

ビューからカジュアルに
Userをcountする


```
users = User.employer
users.each |u|
  u.skills.any? do |skill|
    skill.skill_group.name == "Ruby"
  end
end
```

無駄クエリ

N+1

```
def category_root
  return self if depth == 1

  parent.category_root
end

def developer?
  category_root == "developer"
end
```

無駄クエリ

再帰呼び出しの中でクエリ発行

```
competitions = JobOffer.competitions

competitions.released.each do |c|
  p c.title
end

competitions.opened do |c|
  p c.title
end

competitions.closed.each do |c|
  p c.title
end
```

無駄クエリ

少しずつ違うRelationを何度も呼ぶ

```
journeys.all? do |departure_schedule, destination_schedule|
  departure_schedule.is_punctual? && destination_schedule.is_punctual?
end

def is_punctual?
  (actual_time.to_i - estimate_time.to_i) < service_provider.schedule_buffer
end
```

無駄クエリ
複数の子から呼ばれる親

```
class ArModel < ActiveRecord::Base
  serialize :col1, Array
  serialize :col2, Array
  serialize :col3, Hash
  serialize :col4, Hash
  serialize :col5, Hash
  serialize :col6, Hash
  serialize :col7, Array
  serialize :col8, Array
  serialize :col9, Array
end
```

ARのインスタンス化コスト

serializeの罣

```
# == Schema information
#
# Table name: xxxxxx
#
#      :integer      not null, primary key
#      :integer      not null
#      :string(256)   not null
#      :string(6144)  not null
#      :string(16)    not null
#      :boolean       default(FALSE), not null
#      :string(64)    default(""), not null
#      :float          default(0.0)
#      :float          default(0.0), not null
#      :integer        default(0), not null
#      :float          default(0.0), not null
#      :float
#      :float
#      :boolean        default(TRUE), not null
#      :date
#      :datetime
#      :datetime
#      :datetime
#      :datetime
#      :string(16)     default("draft"), not null
#      :datetime       not null
#      :datetime       not null
#      :integer         default(0), not null
#      :date            not null
#      :boolean         default(FALSE), not null
#      :integer         default(0), not null
#      :integer
#      :text            default(""), not null
#      :boolean         default(FALSE), not null
#      :boolean         default(FALSE), not null
#      :integer         default(0)
#      :integer         default(0)
#      :integer         default(0)
#      :integer         default(0)
#      :text            default(""), not null
#
```

ARのインスタンス化コスト

大量のカラムと不要なインスタンス化


```
Rendered pasokaras/_entry.json.jbuilder (1.6ms)
Rendered pasokaras/_entry.json.jbuilder (1.8ms)
Rendered pasokaras/_entry.json.jbuilder (2.2ms)
Rendered pasokaras/_entry.json.jbuilder (2.0ms)
Rendered pasokaras/_entry.json.jbuilder (1.5ms)
Rendered pasokaras/_entry.json.jbuilder (3.0ms)
Rendered pasokaras/_entry.json.jbuilder (2.5ms)
Rendered pasokaras/_entry.json.jbuilder (2.5ms)
Rendered pasokaras/_entry.json.jbuilder (1.9ms)
Rendered pasokaras/_entry.json.jbuilder (1.6ms)
Rendered pasokaras/_entry.json.jbuilder (1.3ms)
Rendered pasokaras/_entry.json.jbuilder (1.2ms)
Rendered pasokaras/_entry.json.jbuilder (1.3ms)
Rendered pasokaras/_entry.json.jbuilder (1.2ms)
Rendered pasokaras/_entry.json.jbuilder (1.4ms)
Rendered pasokaras/_entry.json.jbuilder (1.3ms)
Rendered pasokaras/_entry.json.jbuilder (1.2ms)
Rendered pasokaras/_entry.json.jbuilder (1.3ms)
Rendered pasokaras/index.json.jbuilder (225.5ms)
Completed 200 OK in 276ms (Views: 241.5ms | ActiveRecord: 3.9ms)
```

過剰なpartial view

jbuilderが割と重い

一つ一つは
数十msの無駄が
積み重なると数百msに

ちなみに

最近で一番やばかったのは

```
class ArModel < ActiveRecord::Base
  serialize :col1, Array
  serialize :col2, Array
  serialize :col3, Hash
  serialize :col4, Hash
  serialize :col5, Hash
  serialize :col6, Hash
  serialize :col7, Array
  serialize :col8, Array
  serialize :col9, Array
end
```

ARのインスタンス化コスト

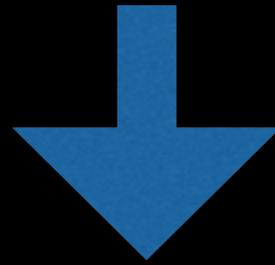
serializeの罣

13.6ms + 0.5ms (1122)
26.7ms + 1.3ms (935)

727.7ms + 31.5ms (3468)

```
def hourly_wage_budget
  if leaf? && metainfo.present? && metainfo.min
    I18n.t("job_category_hourly_wage_buget.#{me
  end
end

def payment_types
  metainfo.payment_types.try(:map, &:to_sym) ||
end
```



12.8ms + 0.5ms (1122)
24.1ms + 0.5ms (935)

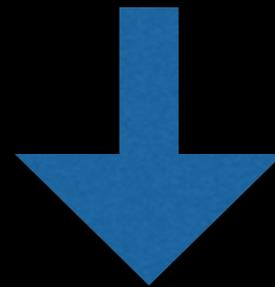
5.8ms + 0.7ms (2312)

```
def hourly_wage_budget
  if leaf? && metainfo.present? && metainfo.min_hourly_wage_
    I18n.t("job_category_hourly_wage_buget.#{metainfo.min_ho
  end
end

def payment_types
  metainfo.payment_types.try(:map, &:to_sym) || []
end
```

39.7ms + 2.6ms (1244)
2181.2ms + 88.8ms (1202)

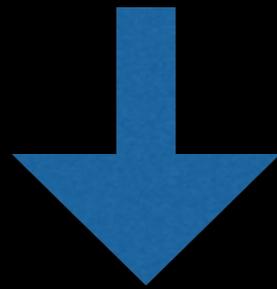
```
def title_helper
  if leaf? && metainfo.present?
    metainfo.title_helper.presence
  end
end
```



36.5ms + 1.5ms (1244)
1.6ms + 0.2ms (601)

```
def title_helper
  if leaf? && metainfo.present?
    metainfo.title_helper.presence || []
  end
end
```

7sec over



1.5sec

無駄と戦うために

DBの 気持ちを知る

**検知と計測のための
ツールを知る**

```
SQL (13.0ms) SELECT `pasokaras`.`id` AS t0_r0, `pasokaras`.`title` AS t0_r1, `pasokaras`.`fullpath` AS t0_r2, `pasokaras`.`nico_
t` AS t0_r5, `pasokaras`.`nico_mylist_count` AS t0_r6, `pasokaras`.`duration` AS t0_r7, `pasokaras`.`nico_description` AS t0_r8, `p
ovie_webm` AS t0_r11, `pasokaras`.`created_at` AS t0_r12, `pasokaras`.`updated_at` AS t0_r13, `tags`.`id` AS t1_r0, `tags`.`name` A
AS t2_r0, `users`.`screen_name` AS t2_r1, `users`.`created_at` AS t2_r2, `users`.`updated_at` AS t2_r3 FROM `pasokaras` LEFT OUTER
aggable_type` = 'Pasokara' LEFT OUTER JOIN `tags` ON `tags`.`id` = `taggings`.`tag_id` LEFT OUTER JOIN `favorites` ON `favorites`.`
tes`.`user_id` WHERE `pasokaras`.`id` IN (57383, 57382, 57381, 57380, 57379, 57378, 57377, 57376, 57375, 57374, 57373, 57372, 57371
359, 57358, 57357, 57356, 57355, 57354, 57353, 57352, 57351, 57350, 57349, 57348, 57347, 57346, 57345, 57344, 57343, 57342, 57341,
SQL (ActiveRecord::Cause) caused by /Users/joker/srcs/seirenes_workspace/seirenes/app/views/pasokaras/index.json.jbuilder:2:in `
58980'
```

activerecord-cause

SQLが実際に発行された場所のバックトレースを出す

- N+1 Query:

```
2009-08-25 20:40:17[INFO] N+1 Query: PATH_INFO: /posts;    model: Post =>
Add to your finder: :include => [:comments]
2009-08-25 20:40:17[INFO] N+1 Query: method call stack:
/Users/richard/Downloads/test/app/views/posts/index.html.erb:11:in `run'
/Users/richard/Downloads/test/app/views/posts/index.html.erb:8:in `each'
/Users/richard/Downloads/test/app/views/posts/index.html.erb:8:in `run'
/Users/richard/Downloads/test/app/controllers/posts_controller.rb:7:in `i
```

bullet

**N+1や使われていないeager load,
counter cacheを検知する**

▼ 応答ヘッダ (0.731 KB)
Cache-Control: "max-age=0, private, must-revalidate"
Connection: "close"
Content-Type: "application/json; charset=utf-8"
Etag: "W/"889fb00ee5dde5767350aa877fdd2592""
Server: "thin"
Set-Cookie: "_seirenes_session=Q1Q5MExhRDZydzd"
X-Content-Type-Options: "nosniff"
X-Frame-Options: "SAMEORIGIN"
X-Request-Id: "2e951459-bcc3-4723-bffb-3841441d4"
X-Runtime: "0.349018"
X-Xss-Protection: "1; mode=block"

rack-runtime

rack層でリクエストにかかった時間を計測して返す

47.1ms + 8.1ms

app/controllers/favorites_controller.rb

```
class FavoritesController < ApplicationController
  before_action :authenticate
  before_action :trim_filter_tags_param, only: [:index]

  def index
    order_by = params[:order_by].try(:split, " ").try(:map, &:to_sym) || [:created_at, :desc]
    search_parameter = Pasokara::Searchable::SearchParameter.new(
      tags: params[:filter_tags],
      page: params[:page],
      order_by: [order_by],
      user_id: current_user.id
    )
    @search = Pasokara.search_with_facet_tags(search_parameter)
    @pasokaras = @search.records.eager_load(:tags, :favorites, :users)
    @facets = @search.response["facets"]["tags"]["terms"]

    render "pasokaras/index"
  end
end
```

0.0ms +	-0.0ms (5)
0.0ms +	0.0ms (2)
0.0ms +	0.0ms (2)
0.0ms +	-0.0ms (2)
0.0ms +	0.0ms (2)
0.4ms +	0.0ms (1)
6.3ms +	8.1ms (2)
0.0ms +	-0.0ms (4)
40.3ms +	0.0ms (1)

rblineprof & peek

アクセスを処理する際に行毎の処理時間を表示する
rack-lineprofもある

StackProf Navigator - JobCategory#payment_types (1 frames)

[← Back to overview](#)

[/Users/joker/crowdworks/crowdworks/app/models/job_category.rb](#)

Callers		
36	30.25%	block (5 levels) in ActionView::CompiledTemplates#_app_views_job_offers__form_html_erb___3785113257911875492_70211746617220
22	18.49%	JobCategory#competition_price_table
20	16.81%	JobCategory#task_num
19	15.97%	JobCategory#competition_expected_proposal_products_table
16	13.45%	JobCategory#task_price
6	5.04%	Delegator#method_missing

Callees		
116	98.31%	block in ActiveRecord::Associations::Builder::Association#define_readers
2	1.69%	block (3 levels) in

Code

```
315   if leaf? && metainfo.present? && metainfo.min_hourly_wage_budget.present? && metainfo.max_hourly_wage_budget.present?  
316     I18n.t("job_category_hourly_wage_buget.#{metainfo.min_hourly_wage_budget}-#{metainfo.max_hourly_wage_budget}")  
317   end  
318 end  
319  
320 def payment_types  
321   metainfo.payment_types.try(:map, &:to_sym) || []  
322 end  
323  
324 def job_offer_types  
325   payment_types.map{ |payment_type| (payment_type == :fixed_price || payment_type == :hourly) ? :project : payment_type }.uniq
```

stackprof-webnav

どの処理がどこから呼ばれて、どれくらいの時間を使っているのか
スタックをどんどん掘って見ていくことができる

ざっくりまとめ

- **DBアクセスの基本を守る**
 - **無駄なクエリを呼ばないこと**
- **cacheに頼るのもいいが根本的に解決できるならその方が良い**
 - **モデル層のパフォーマンスはあらゆる箇所に影響する**
- **serialize(特にYAML)はヤバイから気を付けて使うべし**
- **計測が大事。計測ツールをすぐ使えるように覚えておこう**

最も大切なこと

割れた窓を放置しない