(long edition)

High Performance Rails

Issei Naruta

RubyKaigi 2013

Issei Naruta

@mirakui

VP of DevOps at COOKPAD Inc.

Image delivering &&
site performance improvement

“Speed up Rails”

COOKPAD

COOKPAD is Japan's top recipe site allowing visitors to upload and search through original, user-created recipes.

20 million UU/month
1.38 million recipes

*Jan 2013

COOKPAD

- AWS
- Rails 3.2
- Ruby 2.0.0

Size of Our Rails App

*26 May 2013

- 1003 models
- 236 controllers
- 2871 view templates
- 1978 lines in routes.rb
- 3383 assets in manifest.yml

COOKPAD

200 ms

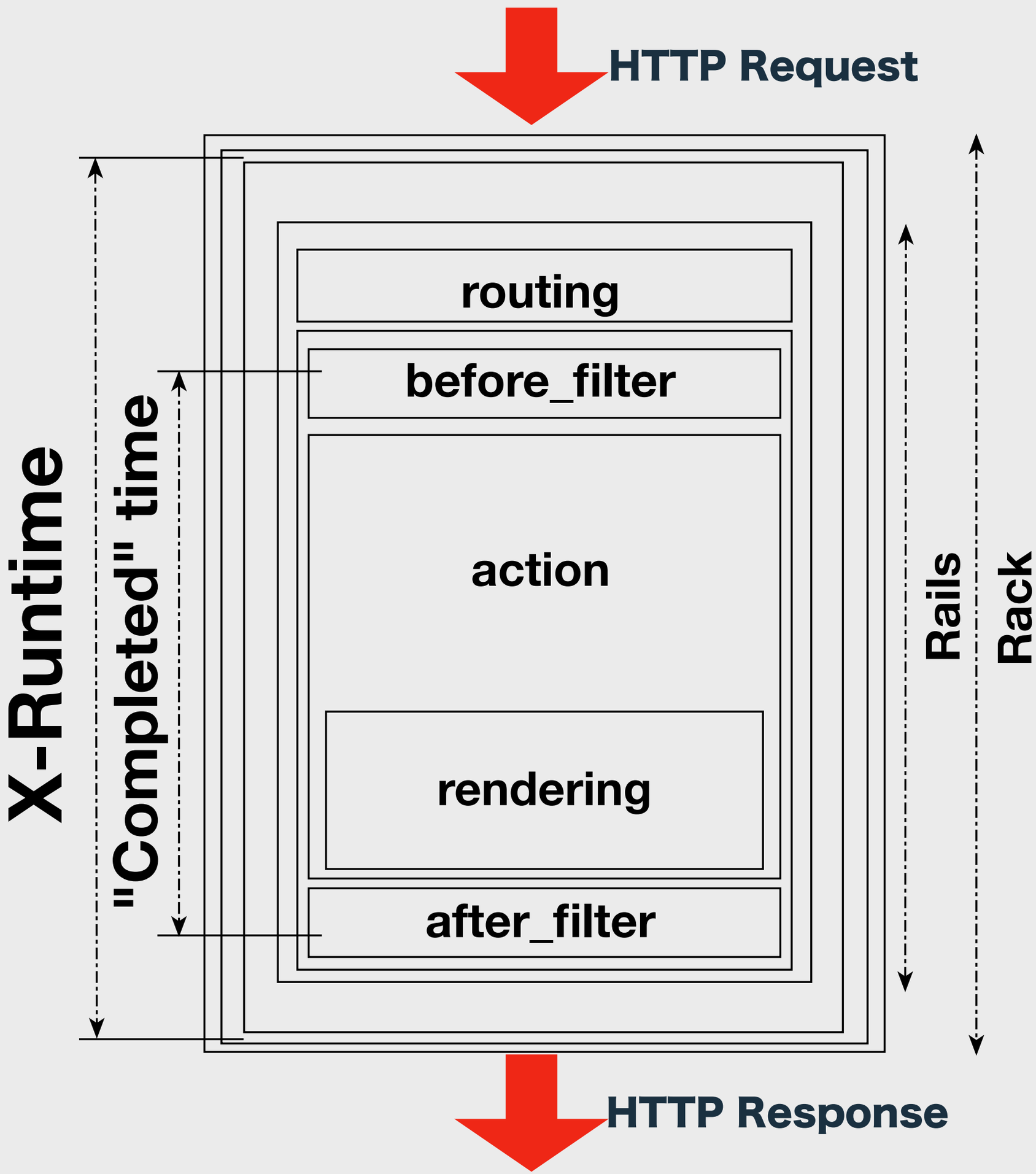
What is the “response time”?

Response time in rails log

```
Started GET "/articles/1" ...
        :
Article Load (0.1ms) SELECT articles.* FROM ...
        :
Rendered articles/show.html.erb within layouts/application (0.7ms)
Completed 200 OK in 100ms (Views: 70.1ms | ActiveRecord: 10.0ms)
```

Response time in response header

```
X-Runtime: 0.123456
```


HTTP Request
routing
before_filter
action
rendering
after_filter
X-Runtime
"Completed" time
Rails
Rack
HTTP Response

Is Rails Slow?

Web Framework Benchmarks

Servlet
(fast)

Rails
(slow)

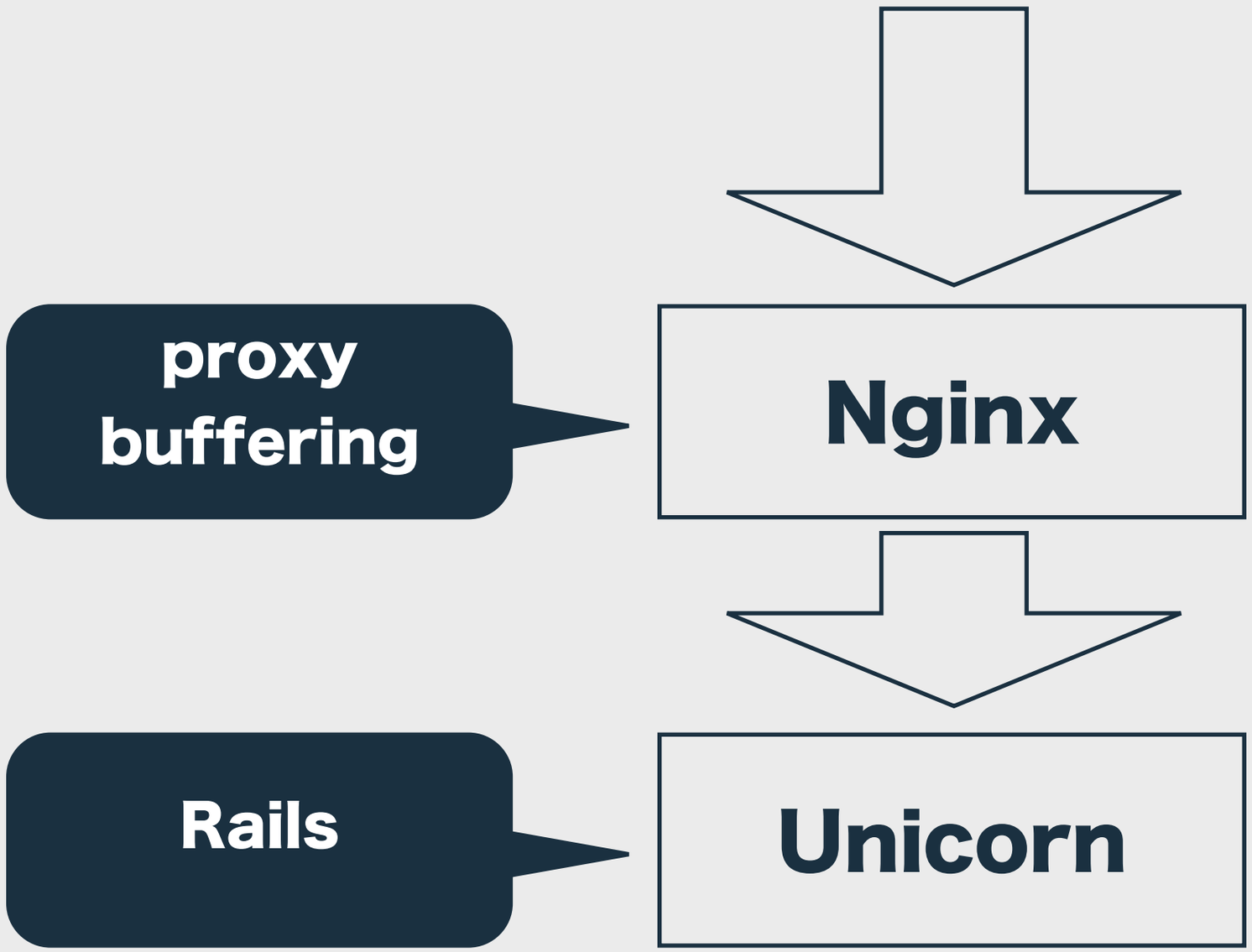
Peak fortunes responses per second, EC2 large

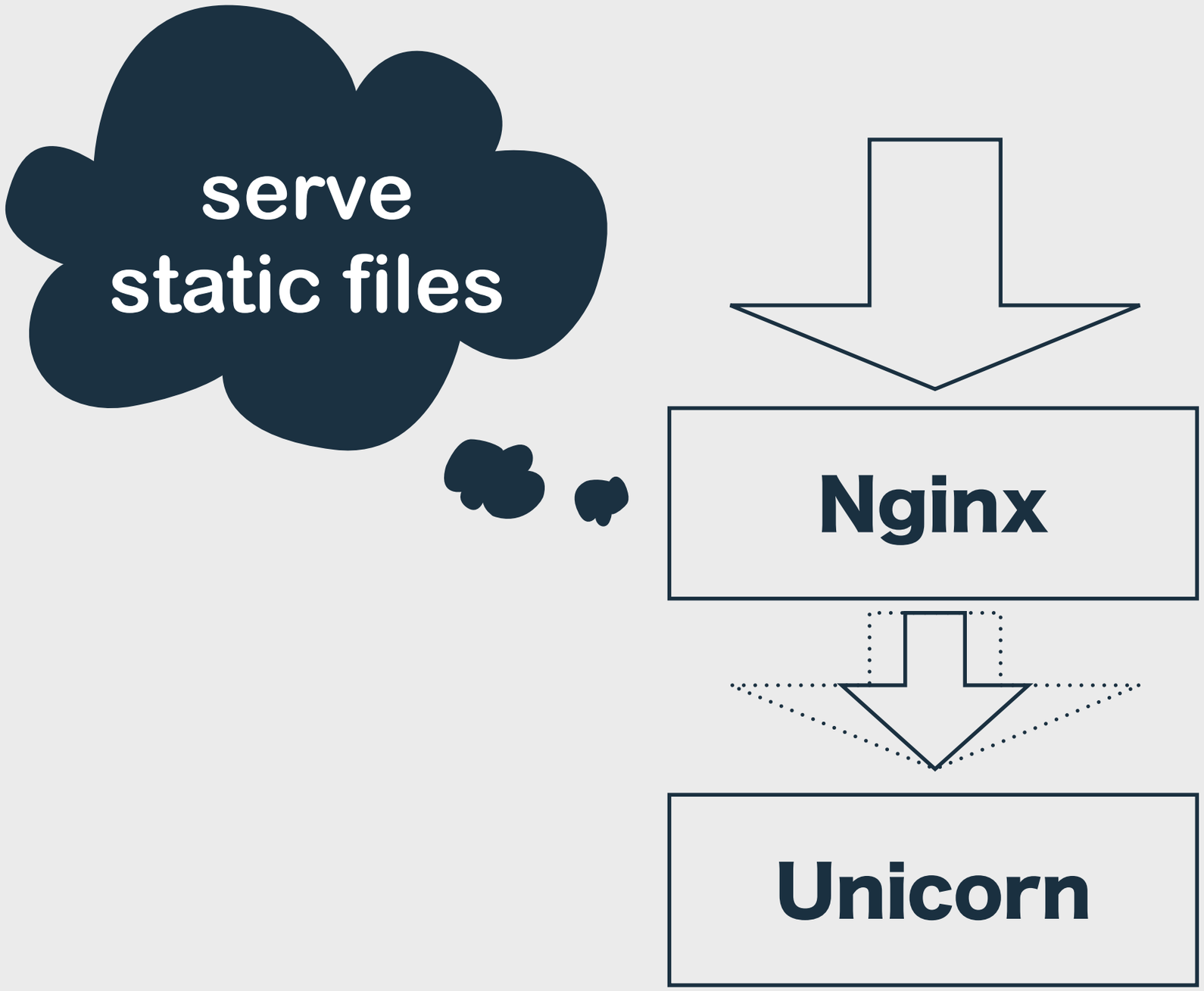
Framework	Peak performance (higher is better)		Cls	Lng	Plt	FE	DB	Orm	IA	Errors
servlet	6,573	100.0%	Plt	Jav	Svt	Res	My	Raw	Rea	155
gemini	5,987	91.1%	Ful	Jav	Svt	Res	My	Mcr	Rea	0
onion	4,520	68.8%	Plt	C	Oni	Non	My	Raw	Rea	142
go	4,121	62.7%	Plt	Go	Go	Non	My	Raw	Rea	0
cpoll-cppsp	4,119	62.7%	Mcr	C++	Cpl	Non	My	Raw	Rea	0
revel	3,502	53.3%	Ful	Go	Go	Non	My	Raw	Rea	0
compojure	3,266	49.7%	Mcr	Clj	Svt	Res	My	Mcr	Rea	0
spring	2,758	42.0%	Ful	Jav	Svt	Res	My	Ful	Rea	0
express	2,454	37.3%	Mcr	JS	njs	Non	My	Ful	Rea	0
play-scala	2,440	37.1%	Ful	Sca	Nty	Non	My	Ful	Rea	0
php	2,312	35.2%	Plt	PHP	FPM	ngx	My	Raw	Rea	0
flask	1,765	26.9%	Mcr	Py	PyP	Non	My	Raw	Rea	0
grizzly	1,758	26.7%	Mcr	Jav	Jty	Res	My	Ful	Rea	0
bottle	1,697	25.8%	Mcr	Py	Wsg	Gun	My	Raw	Rea	0
flask	1,318	20.1%	Mcr	Py	PyP	Non	My	Ful	Rea	0
phalcon	1,264	19.2%	Ful	PHP	FPM	ngx	My	Ful	Str	0
flask	1,210	18.4%	Mcr	Py	Wsg	Gun	My	Raw	Rea	0
ringo	1,050	16.0%	Plt	JS	Rjs	Non	My	Raw	Rea	0
phalcon	806	12.3%	Ful	PHP	FPM	ngx	My	Ful	Rea	0
bottle	733	11.2%	Mcr	Py	Wsg	Gun	My	Ful	Rea	0
ringo	731	11.1%	Plt	JS	Rjs	Non	My	Mcr	Rea	0
flask	682	10.4%	Mcr	Py	Wsg	Uni	My	Ful	Rea	0
rails	483	7.3%	Ful	Rby	JRb	Res	My	Ful	Rea	0
codeigniter	472	7.2%	Ful	PHP	FPM	ngx	My	Raw	Rea	0
django	434	6.6%	Ful	Py	Wsg	Gun	Pg	Ful	Rea	0
rails	398	6.1%	Ful	Rby	Rac	Uni	My	Ful	Rea	0
laravel	225	3.4%	Ful	PHP	FPM	ngx	My	Raw	Rea	0
fuel	177	2.7%	Mcr	PHP	FPM	ngx	My	Raw	Rea	0
symfony2	143	2.2%	Ful	PHP	FPM	ngx	My	Raw	Rea	0
symfony2	109	1.7%	Ful	PHP	FPM	ngx	My	Ful	Rea	0
symfony2	100	1.5%	Ful	PHP	FPM	ngx	My	Ful	Rea	0

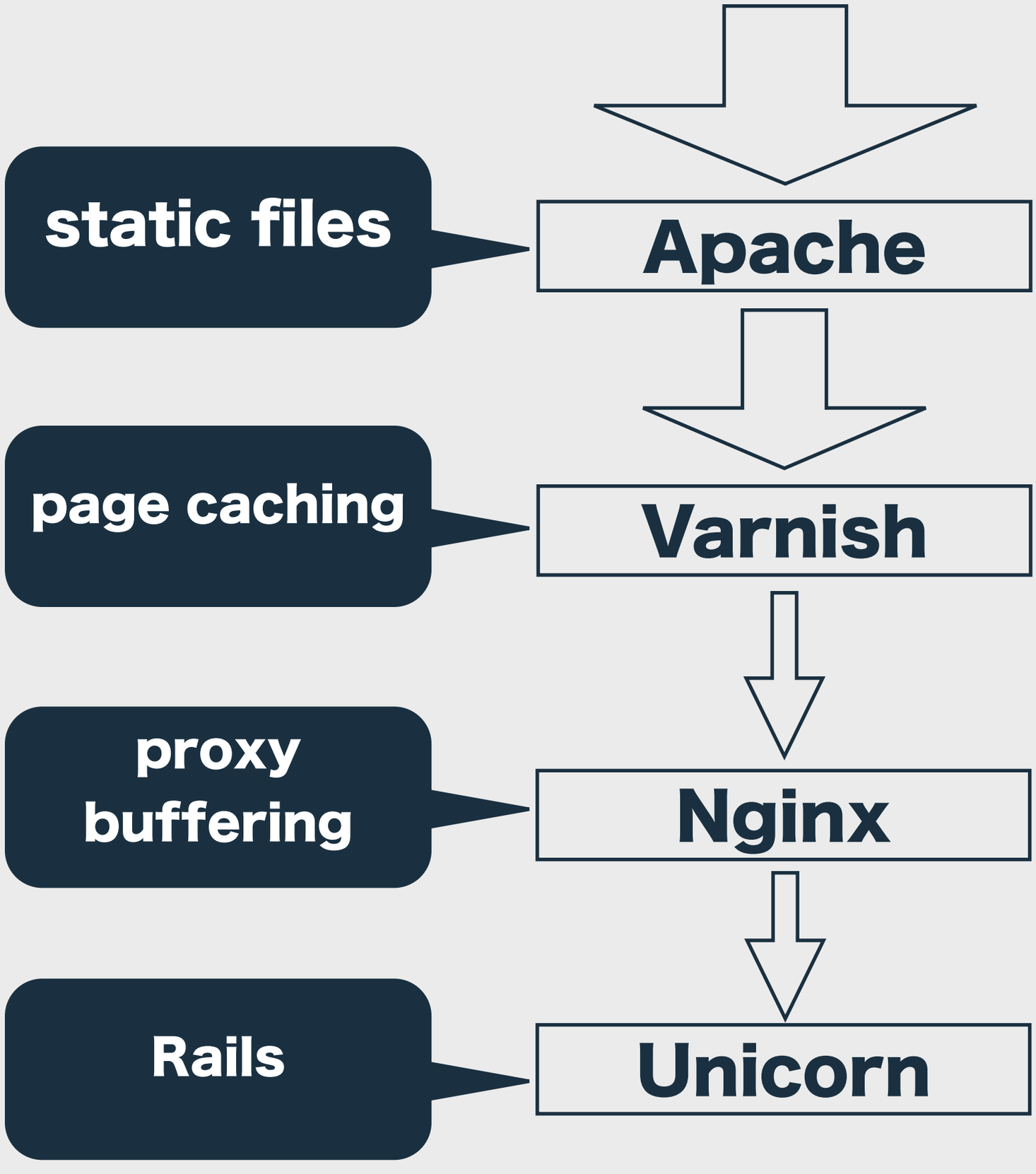
http://www.techempower.com/benchmarks/

Major premise of high performance Rails apps

Don’t process ruby code if possible

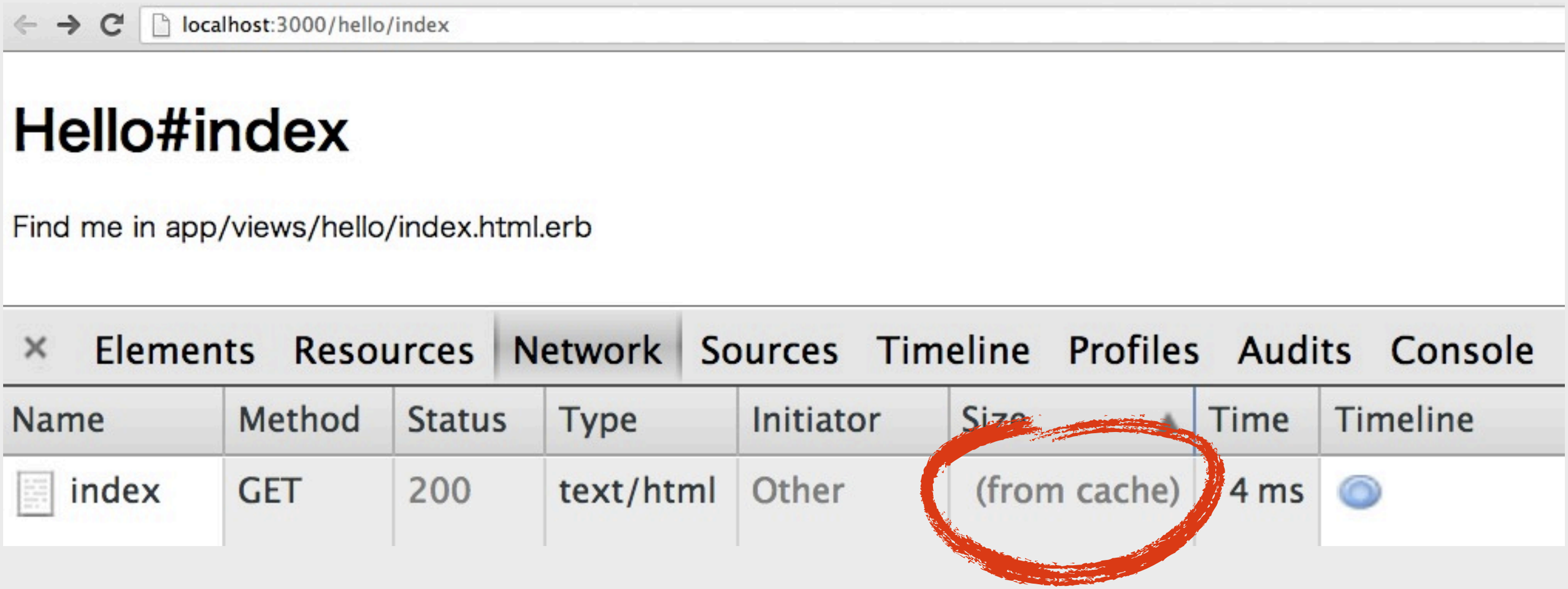

proxy
buffering
Nginx
Rails
Unicorn


serve
static files
Nginx
Unicorn


static files
Apache
page caching
Varnish
proxy
buffering
Nginx
Rails
Unicorn

Browser-side Caching

Browser-side Caching



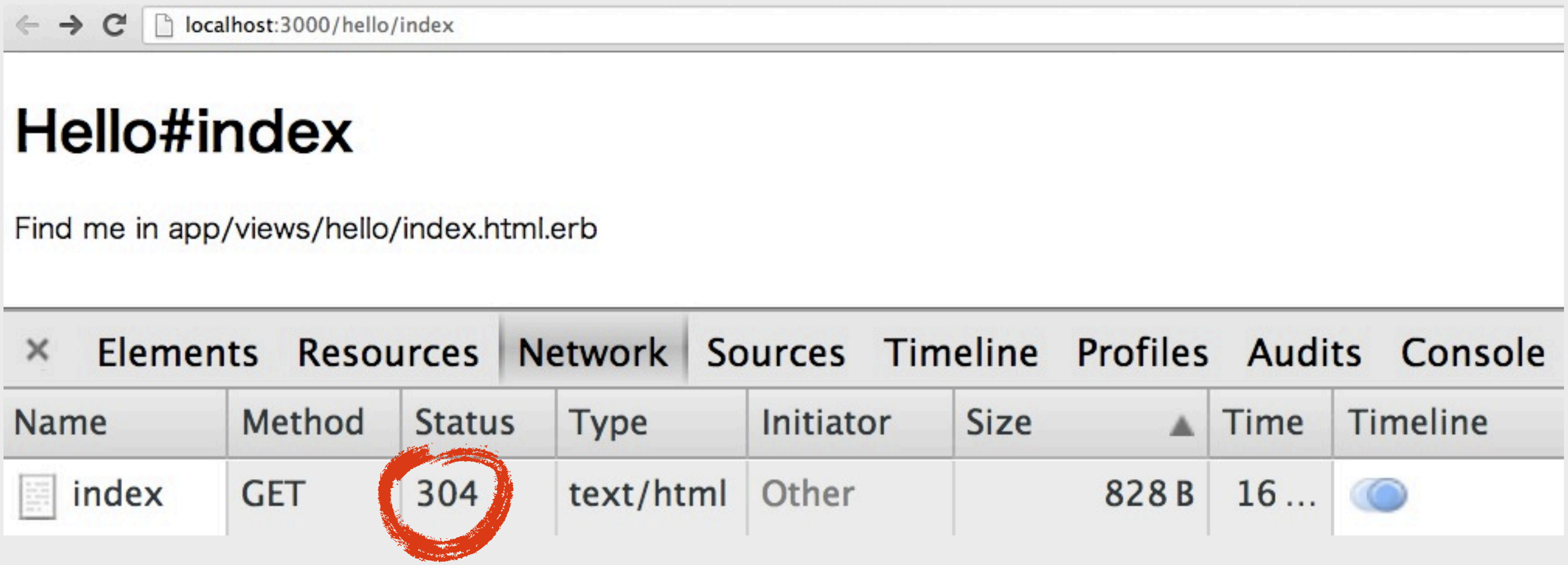
Using browser-side caching

```ruby
def index
end
# => Cache-Control: max-age=0, private, must-revalidate


def index
  expires_in 1.hour
end
# => Cache-Control: max-age=3600, private
```

Never use browser-side caching

```
def index
  expires_now
end
# => Cache-Control: no-cache
```

Conditional GET



1st request

Name	Method	Status	Type	Initiator	Size ▲	Time	Timeline	3
index	GET	200	text/html	Other	1.8 KB	19 ms		

2nd request

Name	Method	Status	Type	Initiator	Size ▲	Time	Timeline	3
index	GET	304	text/html	Other	494 B	15 ms		

```ruby
def index
  @article = Article.last
end
# => 200 OK

def index
  @article = Article.last
  fresh_when @article
end
# => 304 Not Modified
# => Etag: "6a1a7767b6d982df6c53897fd523a761"
# => Last-Modified: Sun, 26 May 2013 01:44:09 GMT
```

fresh_when

- Set `ETag` and `Last-Modified` to response header
- Return “`304 Not Modified`” if;
  - `ETag == If-None-Match`
  - `Last-Modified <= If-Modified-Since`
- Skip rendering if `304` (fast!)

```ruby
def index
  @article = Article.last
  fresh_when @article
end
```

is equal to

```ruby
def index
  @article = Article.last
  fresh_when(
    last_modified: @article.updated_at.utc,
    etag:          @article.cache_key)
end
```

etag { }

(Rails 4)

```ruby
class HelloController < ApplicationController
  etag { current_user.try(:id) }

  def index
    @article = Article.last
    fresh_when @article
  end
end
```

ETag is made of **[@article, current_user.id]**

Ruby Version

Our ruby history

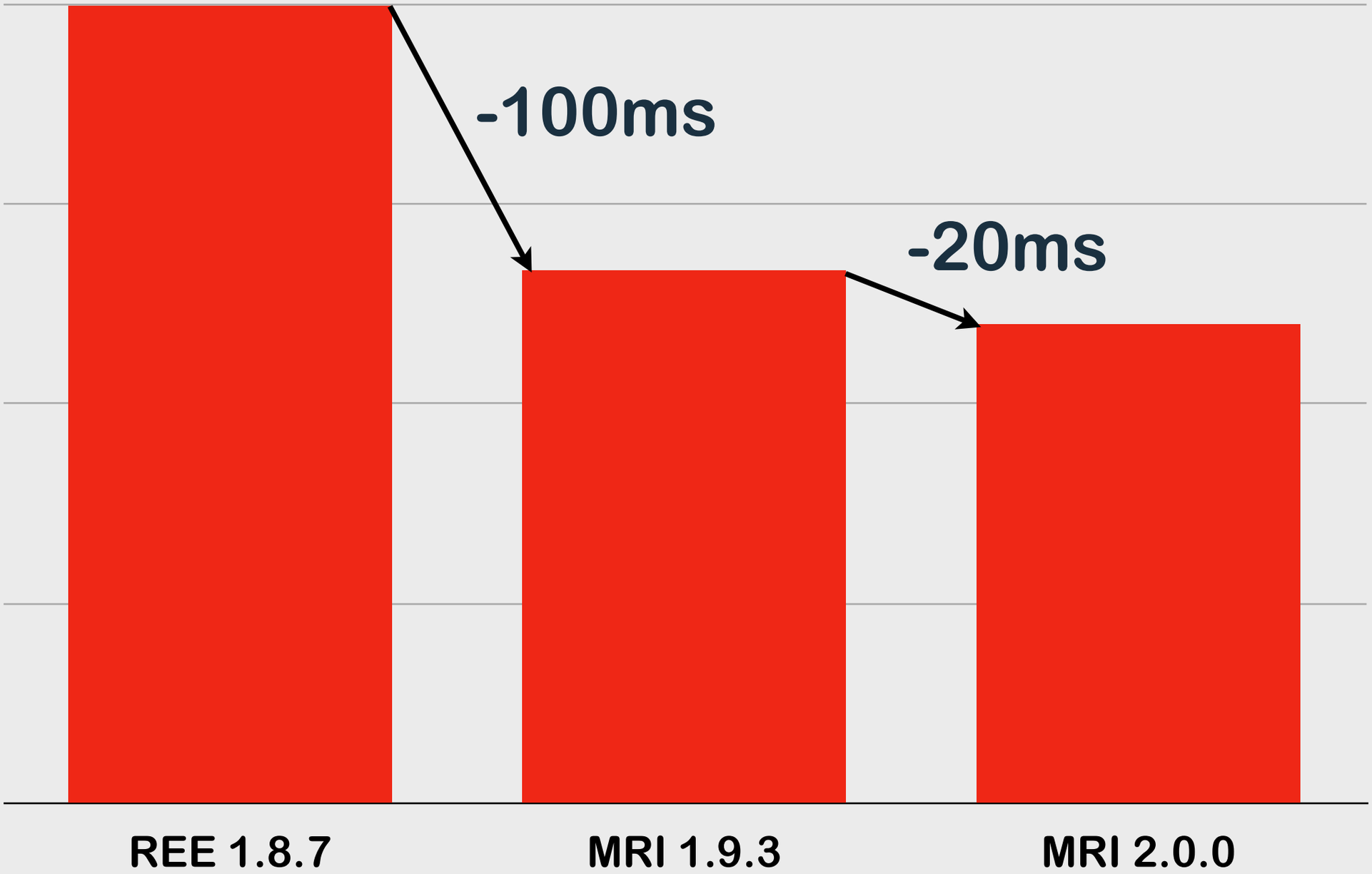
- Sep 2012
  - REE (Ruby Enterprise Edition) 1.8.7
- Feb 2013
  - MRI 1.9.3
- Apr 2013
  - MRI 2.0.0

What's REE

- MRI 1.8.7 + "MBARI patch"
- "100% compatible with MRI 1.8.7"
- Copy-on-Write Friendly GC
- tcmalloc

Response time by ruby versions

COOKPAD



Benchmarks

in this presentation are

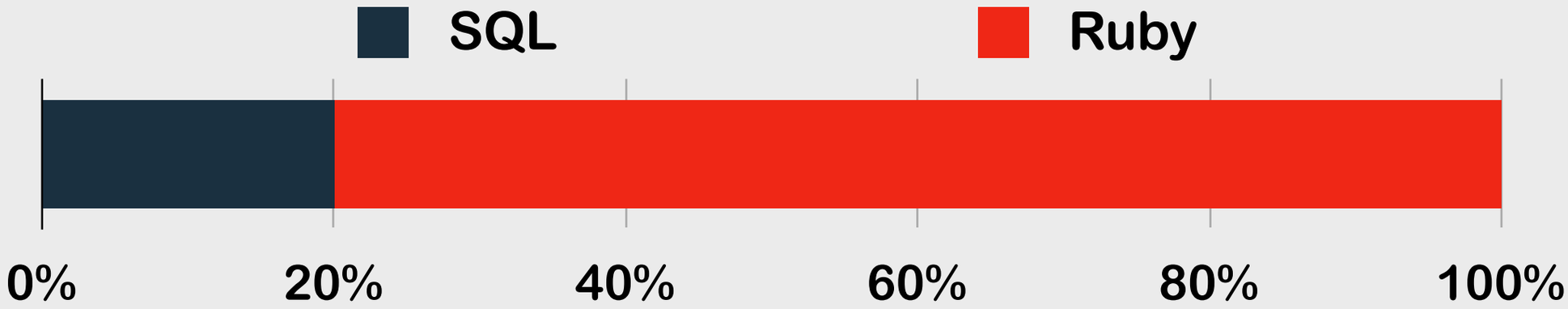
Ruby 2.0.0

Rails 4.0.0 rc1

Performance Bottlenecks in Rails

Response time breakdown

recipe#show (cookpad.com)



Slow stuff

I experienced

- ActiveRecord objects creation
- Resolving routes

ActiveRecord's overhead

```
> Article.last(100).to_a
  Article Load (8.1ms)  SELECT `articles`.* FROM ...
```

ActiveRecord's overhead

```
> Benchmark.ms { Article.last(100).to_a }
Article Load (8.1ms)  SELECT `articles`.* FROM ...
=> 20.27
```

The query is fast enough but creating 100 AR objects is slow

Resolving routes

- Fat routes slow down `url_for`
  - e.g. `link_to` helper

Size of Our Rails App

*26 May 2013

- 1003 models
- 236 controllers
- 2871 view templates
- 1978 lines in routes.rb
- 3383 assets in manifest.yml

Which is faster?

```
link_to 'hello', hello_index_url
link_to ‘hello’, controller: ‘hello’, action: ‘index’
```

Make fat routes

```
# config/routes.rb

get ‘hello/index’, to: ‘hello#index’
1000.times do |i|
  get “hello#{i}”, to: ‘hello#index’
end
```

Result

```
link_to 'hello', hello_index_url
# => 0.09659 ms

link_to ‘hello’, controller: ‘hello’, action: ‘index’
# => 10.97867 ms
```

Slow rack middleware

- Rack::Cache
  - Caches response with Rails.cache
  - Default in Rails 3.2
  - Not default in Rails 4

Caching

Fragment Cache

```
<% cache do -%>
  <%= trend_keywords %>
<% end -%>
```

Default cache key

```
<% cache do -%>
```

controller

cache digest (Rails4)

```
views/localhost:3000/hello/cache_test/3e9258928c27f4ffc5135520c3a976c5
```

action

Fragment cache key

```
<% cache fragment: 'key1' do %>
```

fragment key

```
views/localhost:3000/hello/cache_test?fragment=key1
/3e9258928c27f4ffc5135520c3a976c5
```

Global cache key

```
<% cache 'key1' do %>
```

global key

views/**key1**/116839051556390fb4d5b25362cfe6eb

AR cache key

```
<% cache @article do %>
```

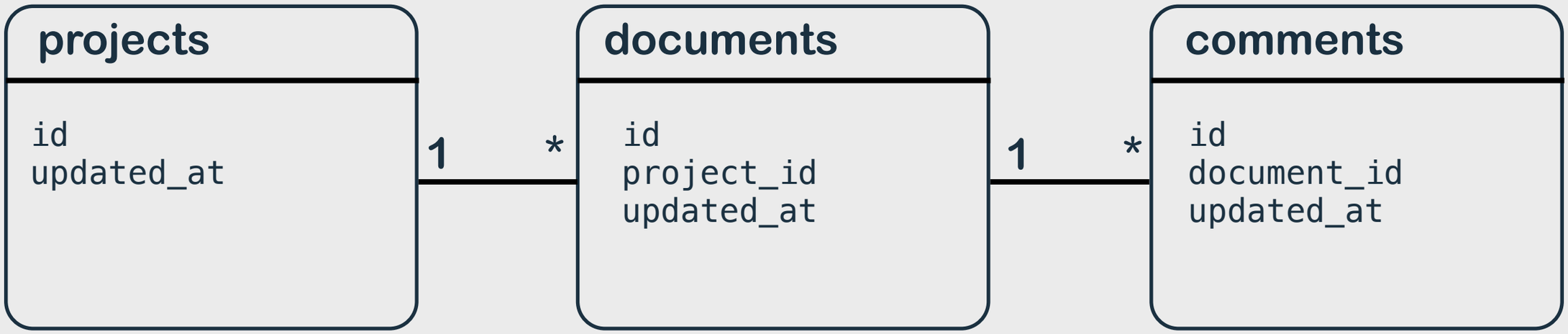
table name

@article.cache_key
(id, updated_at)

```
views/articles/1-20130527190532189241000
/116839051556390fb4d5b25362cfe6eb
```

Cache Digests

(a.k.a. “Russian-doll caching”)




```
has_many :documents
```

```
has_many :comments
belongs_to :document
```

```
belongs_to :documents
```

projects/show.html.erb

cache project do

_document.html.erb

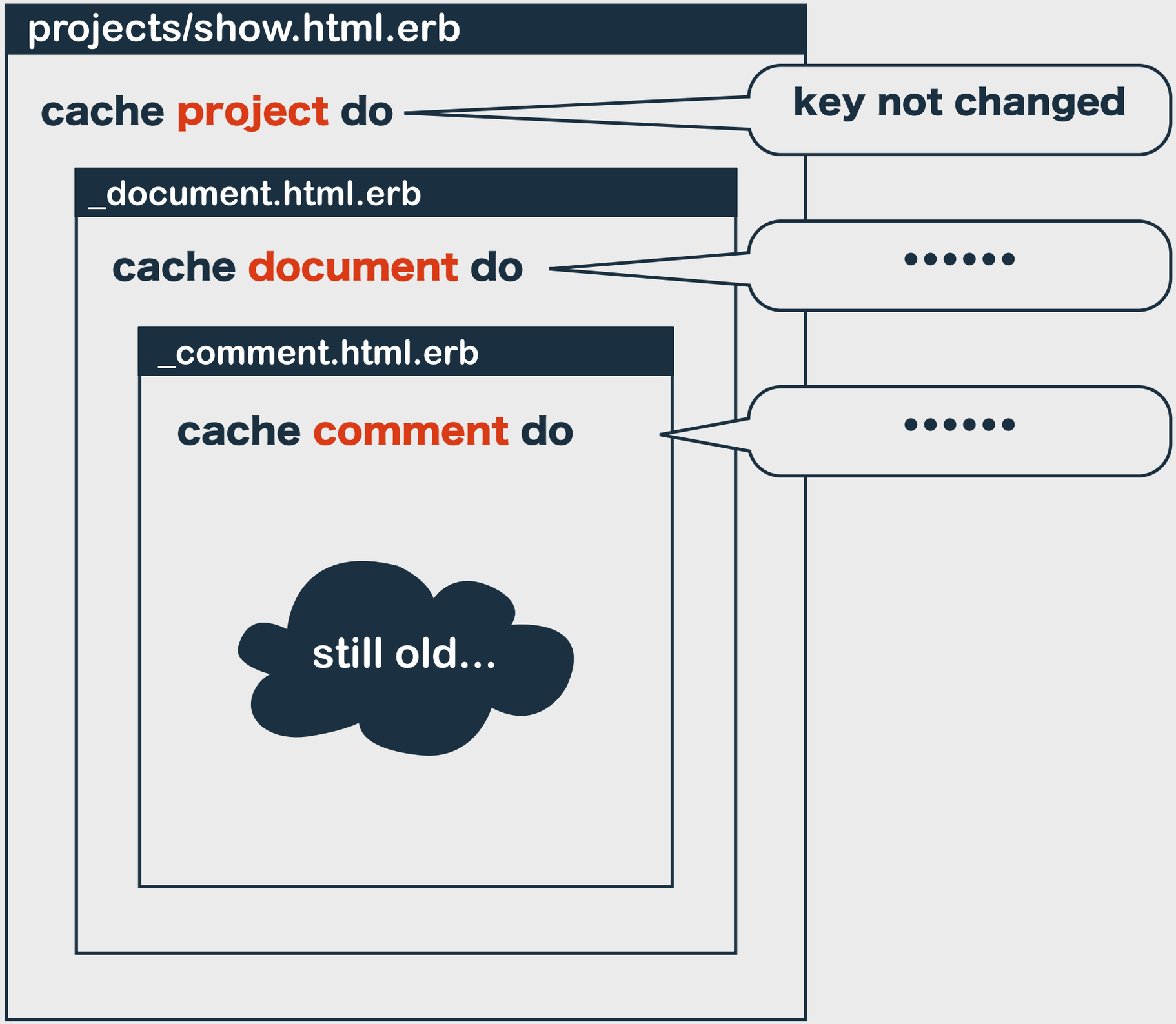
cache document do

_comment.html.erb

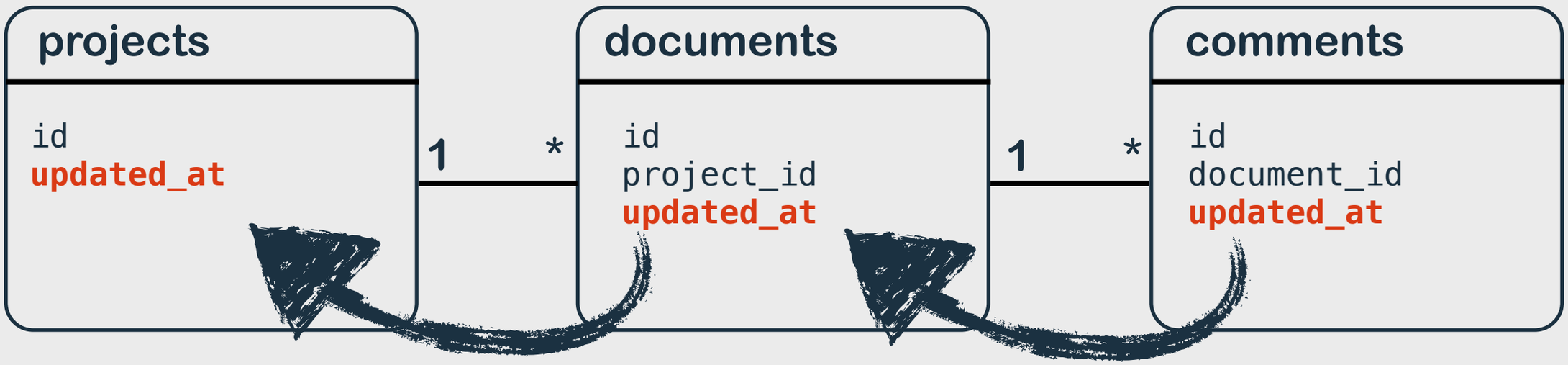
cache comment do

comment.update!

Rails 3


projects/show.html.erb
cache project do
key not changed
_document.html.erb
cache document do
••••••
_comment.html.erb
cache comment do
••••••
still old...

update

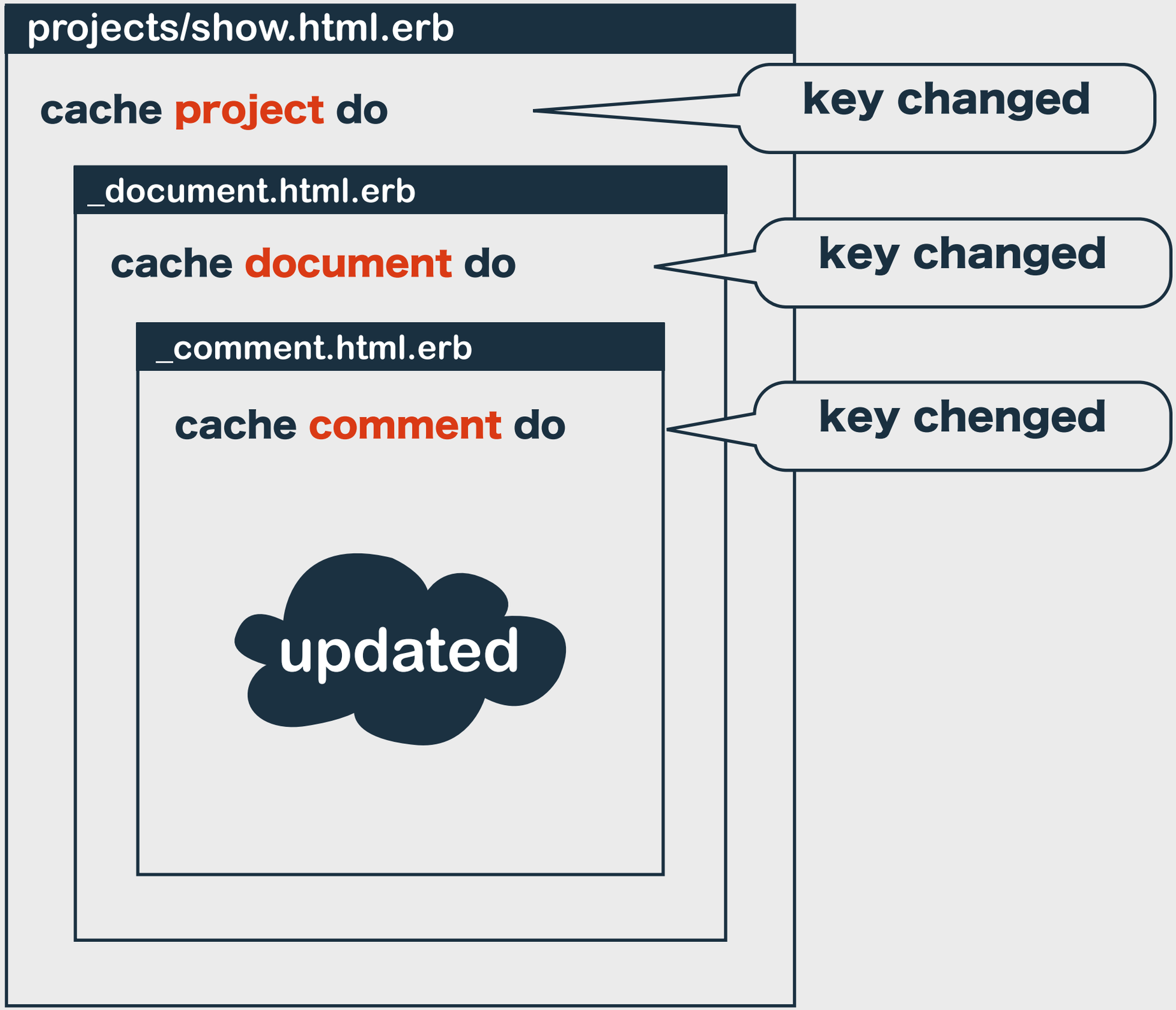



```
has_many :documents
```

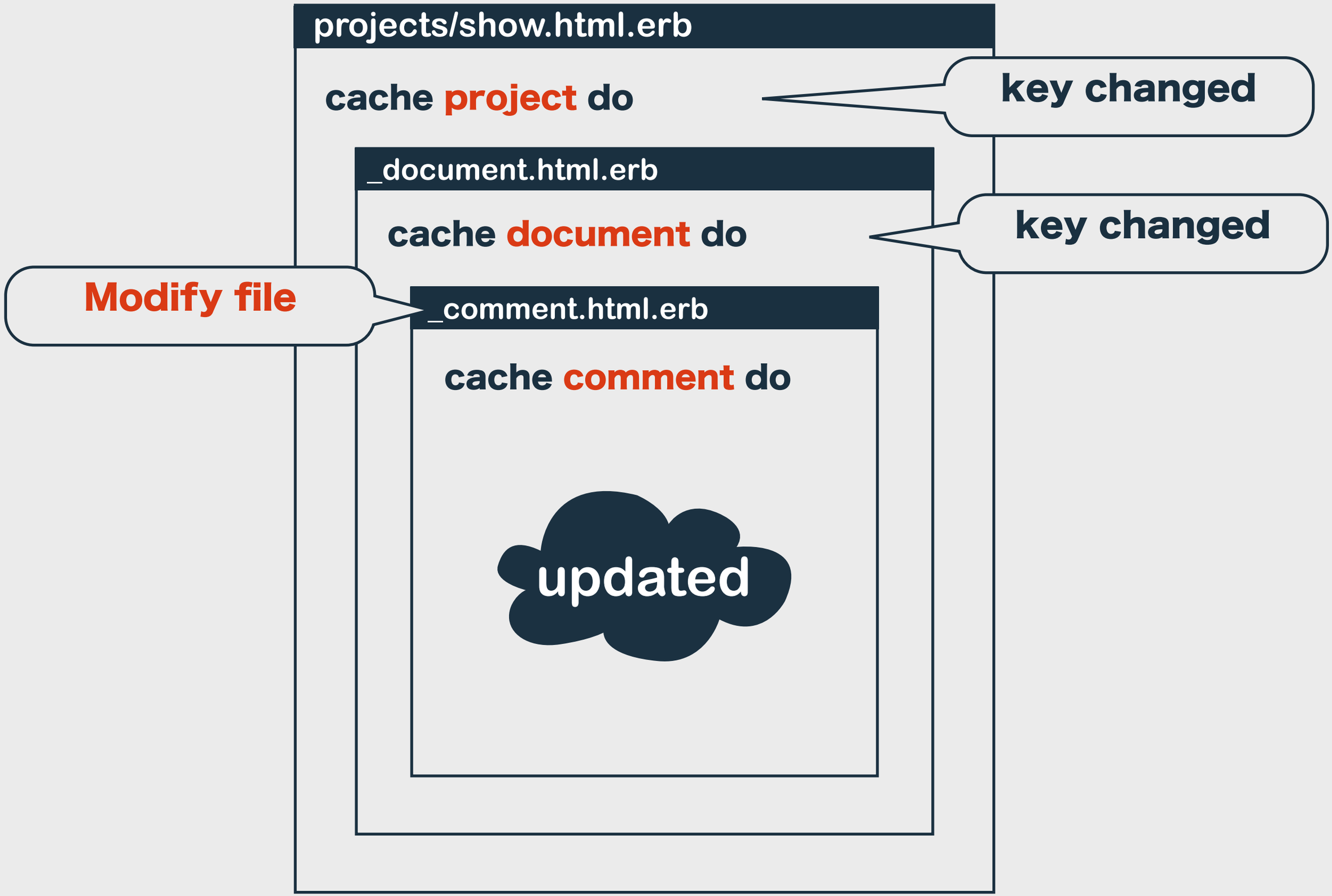
```
has_many :comments
belongs_to :document,
           touch: true
```

```
belongs_to :documents,
           touch: true
```

Rails 4 Way


projects/show.html.erb
cache project do
key changed
_document.html.erb
cache document do
key changed
_comment.html.erb
cache comment do
key chenged
updated

Rails 4 Way


projects/show.html.erb
cache project do
key changed
_document.html.erb
cache document do
key changed
Modify file
_comment.html.erb
cache comment do
updated

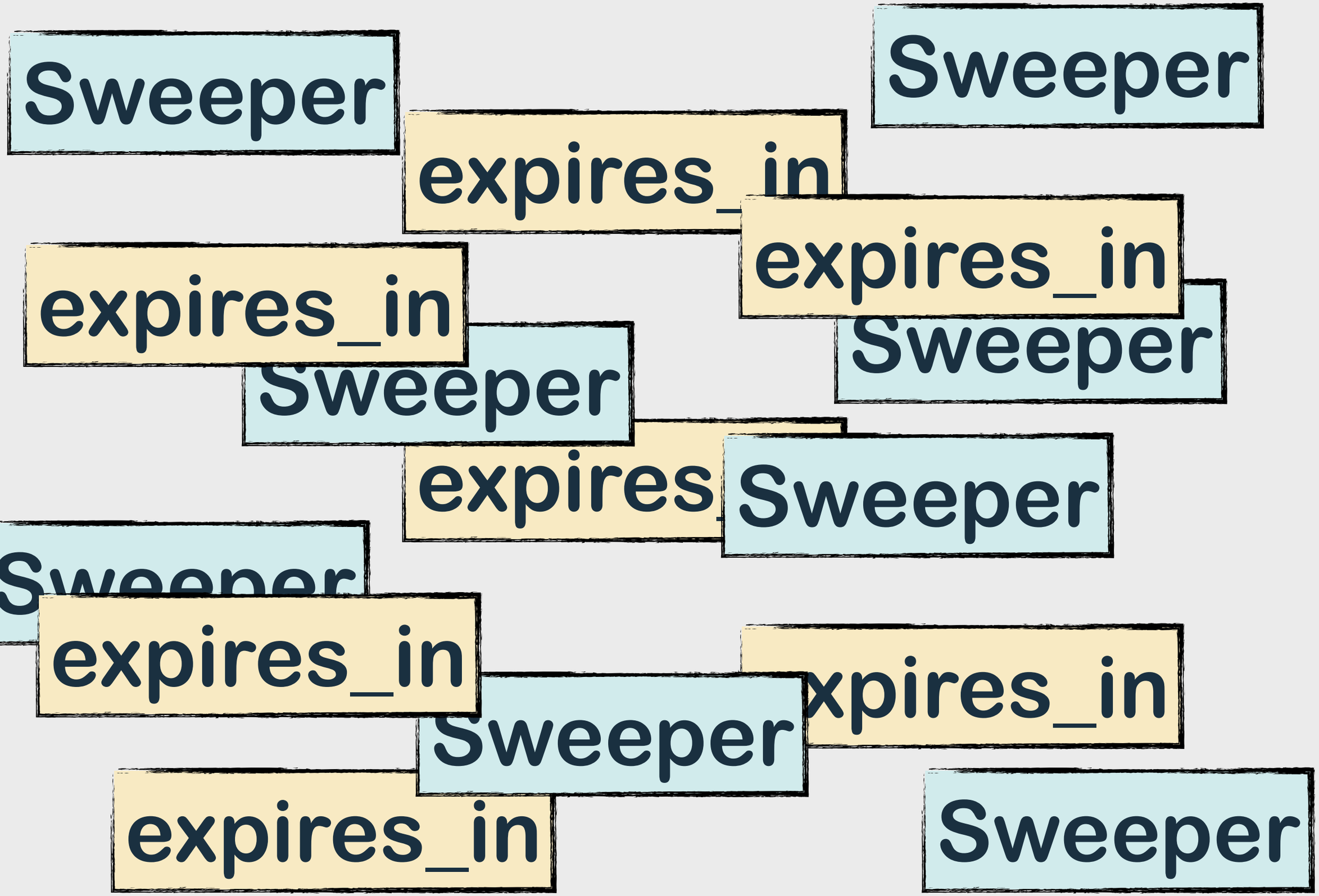
made of [id, updated_at]

views/documents/1-2013052719053218924100 0/1168390515563 90fb4d5b25362cfe6eb

MD5 of the template file itself
and all of its dependencies

Action Cache

```
caches_action :show,
              if: -> { !current_user.staff? },
              cache_path: -> { custom_cache_path },
              expires_in: 1.hour
```


Sweeper
Sweeper
expires_in
expires_in
expires_in
Sweeper
Sweeper
expires_in
Sweeper
expires_in
Sweeper
Sweeper
expires_in
xpires_in
Sweeper
expires_in
Sweeper

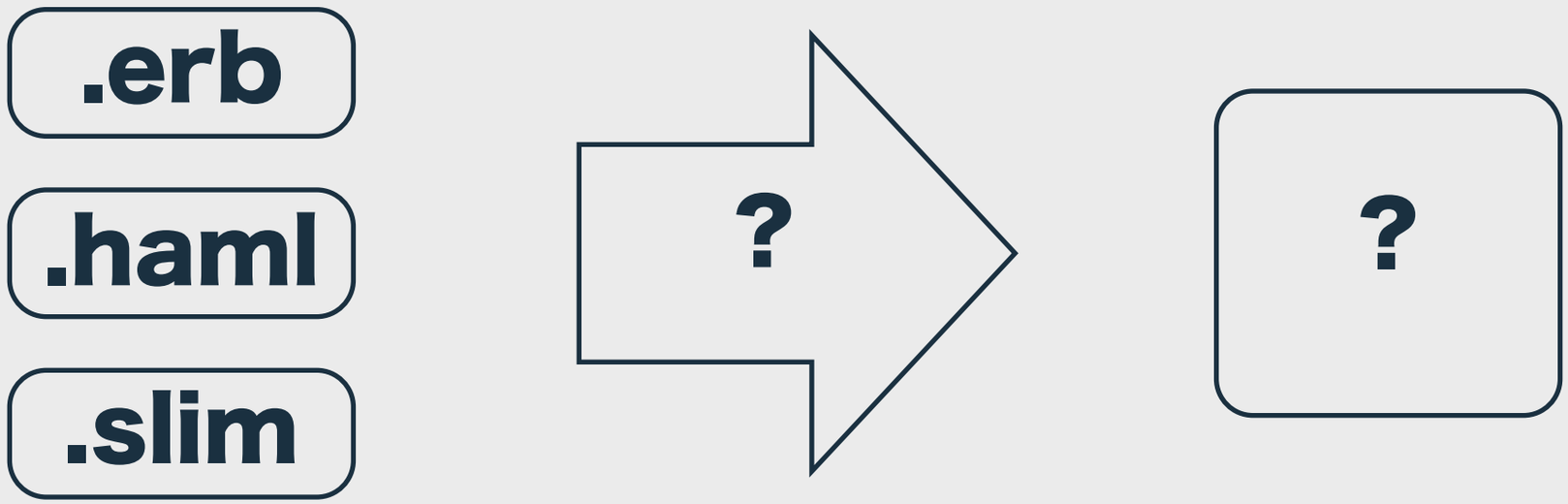
Action Caching is not Rails 4 Way

Use Russian-doll Caching

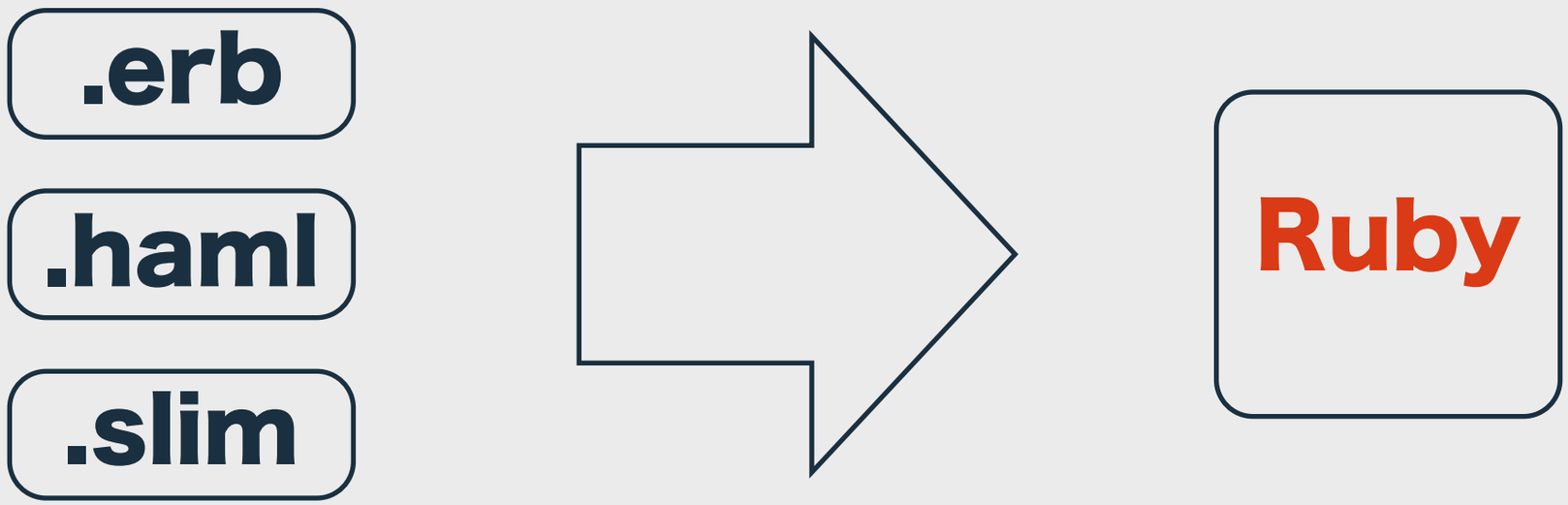
Rails 3	Extracted as gems for Rails 4
action cache	actionpack-action_caching
page cache	actionpack-page_caching
sweeper	rails-observers

Template Engines

How Rails render templates



How Rails render templates



ActionView::Template#compile
at first time of
rendering each templates

ERB (erubis)

```erb
<ul>
  <% 100.times do |i| -%>
    <li><%= i %></li>
  <% end -%>
</ul>
```

```ruby
module ActionView::CompiledTemplates
  def _app_views_hello_erb_test_html_erb__74102564489664903 5_70151266816280(local_assigns, output_buffer)
    _old_virtual_path, @virtual_path = @virtual_path, "hello/erb_test"
    _old_output_buffer = @output_buffer

    @output_buffer = output_buffer || ActionView::OutputBuffer.new
    @output_buffer.safe_append='<ul>'
    100.times do |i|
      @output_buffer.safe_append='    <li>'
      @output_buffer.append=( i )
      @output_buffer.safe_append='</li>'
    end
    @output_buffer.safe_append='</ul>'
    @output_buffer.to_s
  ensure
    @virtual_path, @output_buffer = _old_virtual_path, _old_output_buffer
  end
end
```

Slim

```
ul
  - 100.times do |i|
    li= i
```

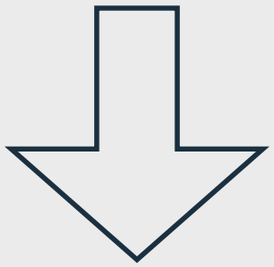
```
module ActionView::CompiledTemplates
  def _app_views_hello_slim_test_html_slim___1819064935872341951_703210610924 80(local_assigns, output_buffer)
    _old_virtual_path, @virtual_path = @virtual_path, "hello/slim_test"
    _old_output_buffer = @output_buffer

    @output_buffer = output_buffer || ActiveSupport::SafeBuffer.new
    @output_buffer.safe_concat(("<ul>"))
    100.times do |i|
      @output_buffer.safe_concat(("<li>"))
      @output_buffer.safe_concat(((::Temple::Utils.escape_html_safe((i))).to_s))
      @output_buffer.safe_concat(("</li>"))
    end
    @output_buffer.safe_concat(("</ul>"))
    @output_buffer
  ensure
    @virtual_path, @output_buffer = _old_virtual_path, _old_output_buffer
  end
end
```

Haml

```
%ul
  - 100.times do |i|
    %li= i
```

```
module ActionView::CompiledTemplates
  def _app_views_hello_haml_test_html_haml__2896236196440990407_70321060660000(local_assigns, output_buffer)
    _old_virtual_path, @virtual_path = @virtual_path, "hello/haml_test"
    _old_output_buffer = @output_buffer

    begin
      extend Haml::Helpers
      _hamlout = @haml_buffer = Haml::Buffer.new(haml_buffer, {
        :autoclose=>["meta", "img", "link", "br", "hr", "input", "area", "param", "col", "base"],
        :preserve=>["textarea", "pre", "code"],
        :attr_wrapper=>"'", :ugly=>false, :format=>:html5, :encoding=>"UTF-8",
        :escape_html=>true, :escape_attrs=>true, :hyphenate_data_attrs=>true, :cdata=>false})
      _erbout = _hamlout.buffer
      @output_buffer = output_buffer ||= ActionView::OutputBuffer.new rescue nil
      _hamlout.push_text("<ul>\n", 1, false)

      100.times do |i|
        _hamlout.push_text("  <li>#{
          _hamlout.adjust_tabs(1)
          _hamlout.format_script_false_true_false_true_false_true_false((i))
        }</li>\n", 0, false)
      end
      _hamlout.push_text("</ul>\n", -1, false)
      ::Haml::Util.html_safe(_erbout)
    ensure
      @haml_buffer = @haml_buffer.upper if @haml_buffer
    end

  ensure
    @virtual_path, @output_buffer = _old_virtual_path, _old_output_buffer
  end
end
```

render ‘erb_test’



Calls

_app_views_hello_erb_test_html_erb__7410256448966490 35_70151266816280

Compiling time is not critical matter for Rails apps, but rendering time is.

Benchmark

ERB (erubis)

```
<ul>
  <% 100.times do |i| -%>
    <li><%= i %></li>
  <% end -%>
</ul>
```

Haml

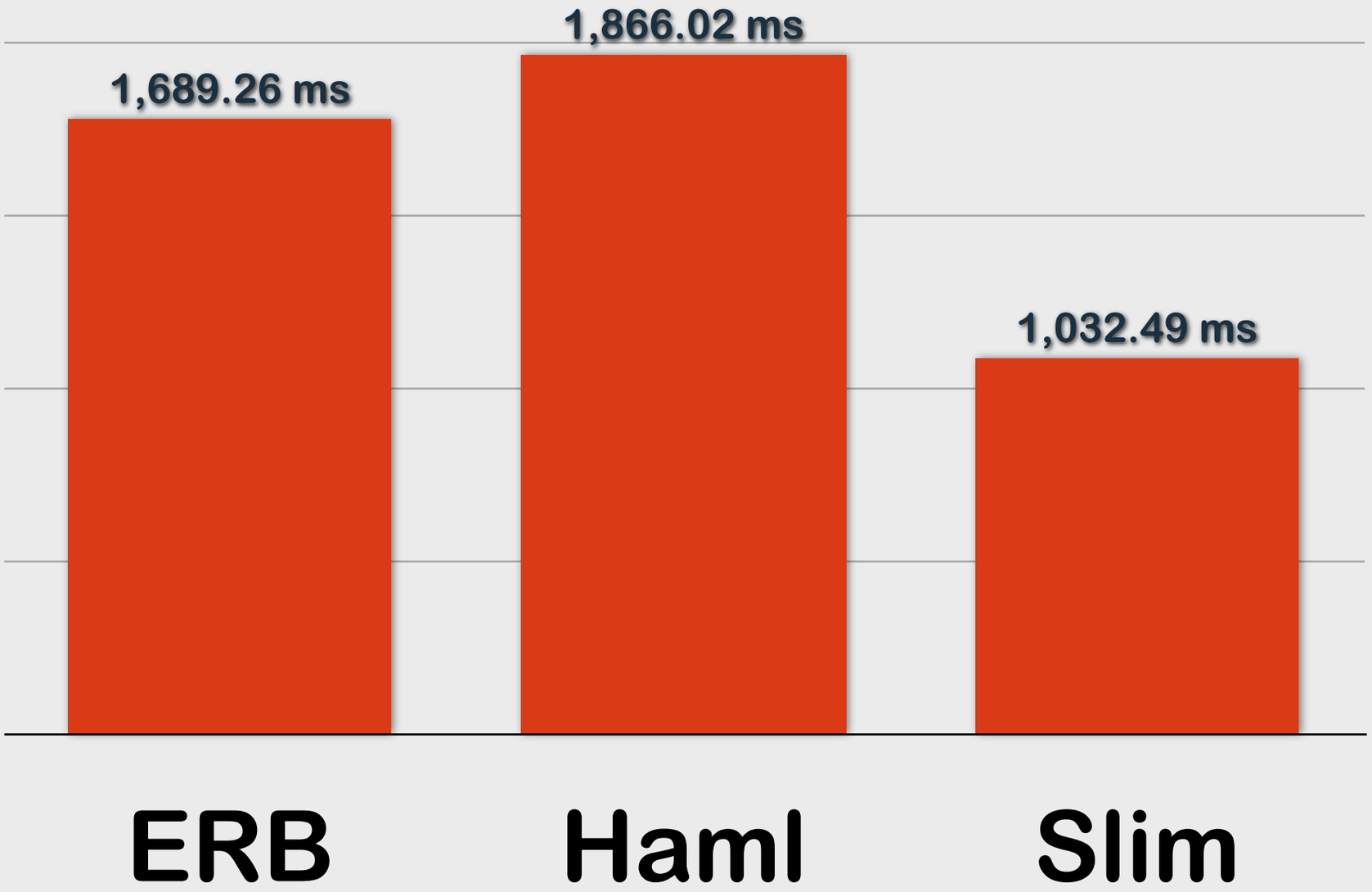
```
ul
  - 100.times do |i|
    li= i
```

Slim

```
%ul
  - 100.times do |i|
    %li= i
```

```
1000.times do
  render_to_string
end
```

Rendering Time



Unicorn with GC.disable

HTTP Requests

Unicorn Default

GC.disable

Run GC each 10 requests

```
# config.ru
require “unicorn/oob_gc”
use Unicorn::OobGC, 10
```

```
# unicorn config

after_fork do |server, worker|
  GC.disable
end
```

unicorn-worker-killer

https://github.com/kzk/unicorn-worker-killer

```
# config.ru

# Unicorn self-process killer
require 'unicorn/worker_killer'

# Max requests per worker
use Unicorn::WorkerKiller::MaxRequests, 3072, 4096

# Max memory size (RSS) per worker
use Unicorn::WorkerKiller::Oom, (192*(1024**2)),
(256*(1024**2))
```

Profiling

ruby-prof

Profile Report

Thread ID	Total Time
70218911272160	1.1192749999999996

Thread 70218911272160

%Total	%Self	Total	Self	Wait	Child	Calls	Name
100.00%	**0.01%**	**1.12**	**0.00**	**0.00**	**1.12**	**2**	**Global#[No method]**
		1.12	0.00	0.00	1.12	1/8	Kernel#load
		0.00	0.00	0.00	0.00	7/8	ActiveSupport::Dependencies::Loadable#load_d
		1.12	0.00	0.00	1.12	1/8	Global#[No method]
99.99%	**0.05%**	**1.12**	**0.00**	**0.00**	**1.12**	**8**	***Kernel#load**
		1.12	0.00	0.00	1.12	2/1124	Kernel#require
		0.04	0.00	0.00	0.04	1/1	ActionDispatch::Routing::RouteSet#draw
		0.00	0.00	0.00	0.00	16/1350	IO#set_encoding
		0.00	0.00	0.00	0.00	2/110	<Class::File>#expand_path
		0.00	0.00	0.00	0.00	2/65	Rails::Railtie::Configurable::ClassMethods#config
		0.00	0.00	0.00	0.00	2/33	<Module::ActiveSupport>#on_load
		0.00	0.00	0.00	0.00	1/3	Rails::Application::Configuration#session_store
		0.00	0.00	0.00	0.00	1/2	Rails::Railtie::Configurable::ClassMethods#meth
		0.00	0.00	0.00	0.00	594/1124	ActiveSupport::Dependencies::Loadable#load_d
		0.00	0.00	0.00	0.00	492/1124	Kernel#require
		0.00	0.00	0.00	0.00	1/1124	ActiveSupport::XmlMini#backend=
		0.00	0.00	0.00	0.00	1/1124	<Class::Rails::Railtie>#inherited
		0.00	0.00	0.00	0.00	1/1124	Rails::Railtie#config
		0.00	0.00	0.00	0.00	6/1124	Array#each
		0.00	0.00	0.00	0.00	14/1124	<Module::Kernel>#require

Thread: 70098257923280 (100.00% ~ 1.1408420000000001)

- 100.00% (100.00%) Global#[No method] [1 calls, 2 total]
  - 99.99% (99.99%) Kernel#load [1 calls, 8 total]
    - 99.94% (99.95%) Kernel#require [2 calls, 1124 total]
      - 99.83% (99.89%) Kernel#require [4 calls, 1124 total]
        - + 59.97% (60.07%) Rails::Application#require_environment! [1 calls, 1 total]
        - 39.53% (39.60%) Kernel#require [3 calls, 1124 total]
          - 25.39% (64.23%) Kernel#require [2 calls, 1124 total]
            - 14.04% (55.28%) Array#each [1 calls, 821 total]
              - 14.03% (99.98%) Kernel#require [6 calls, 1124 total]
                - 13.49% (96.15%) Kernel#require [10 calls, 1124 total]
                  - 10.25% (75.97%) Kernel#require [50 calls, 1124 total]
                    - 8.66% (84.45%) Kernel#require [48 calls, 1124 total]
                      - 6.21% (71.74%) Kernel#require [65 calls, 1124 total]
                        - 2.05% (33.06%) Kernel#require [19 calls, 1124 total]
                          - 1.39% (67.52%) Kernel#require [16 calls, 1124 total]
            - 11.31% (44.53%) Kernel#require [1 calls, 1124 total]
              - 11.23% (99.37%) Kernel#require [10 calls, 1124 total]
                - 10.10% (89.88%) Kernel#require [14 calls, 1124 total]
                  - 8.48% (83.97%) Kernel#require [19 calls, 1124 total]
                    - 6.12% (72.17%) Kernel#require [30 calls, 1124 total]
                      - 3.02% (49.41%) Kernel#require [12 calls, 1124 total]
                        - 2.14% (70.87%) Kernel#require [15 calls, 1124 total]
          - 13.11% (33.17%) <Module::Bundler>#require [1 calls, 1 total]
            - 13.11% (99.99%) Bundler::Runtime#require [1 calls, 1 total]

ruby-prof

- Call-tree is too complex to profile Rails apps

ActiveSupport::Benchmarkable

(Rails 3.2 or later)

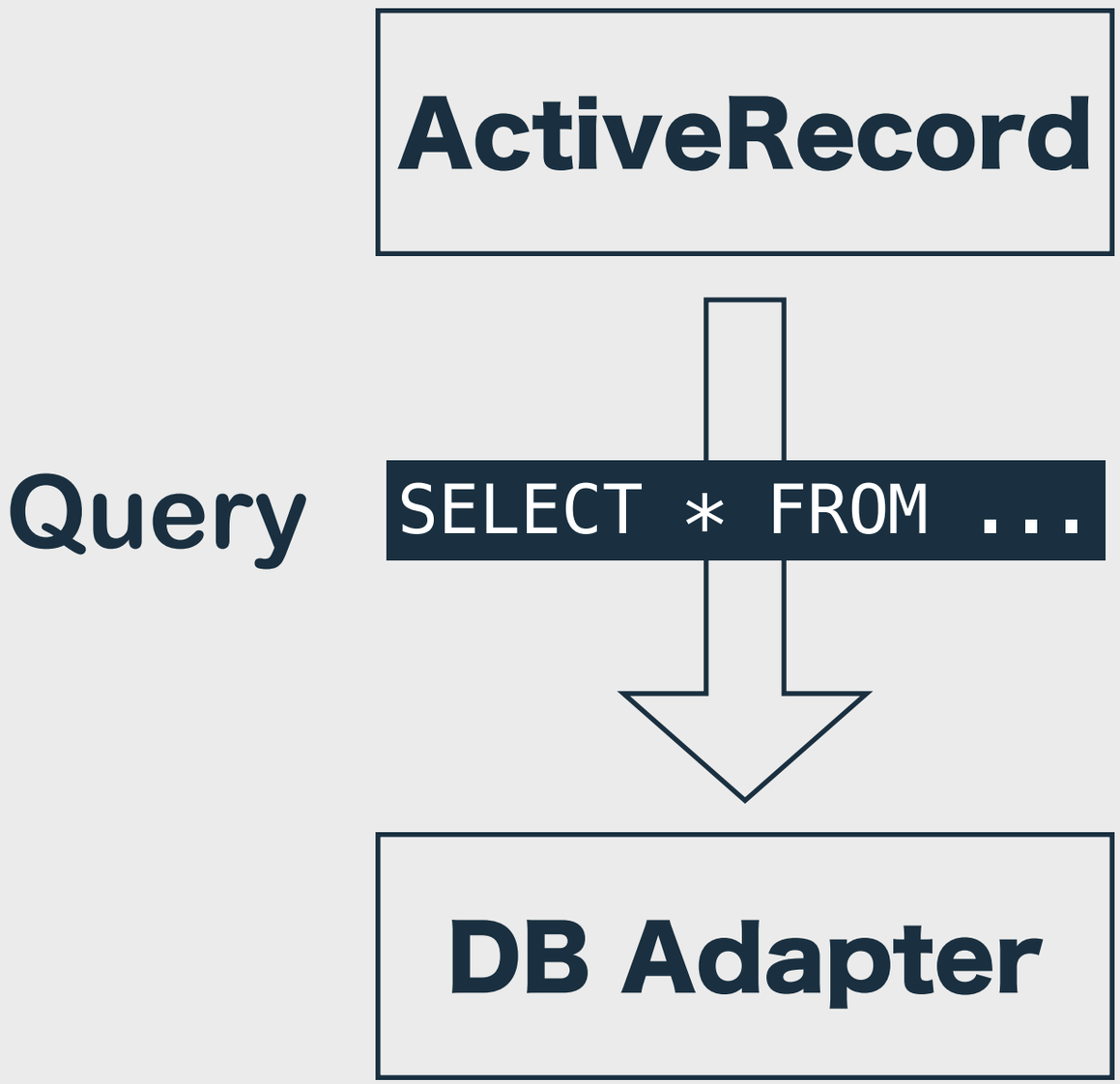
```
<% benchmark "Process data files" do %>
  <%= expensive_files_operation %>
<% end %>
```

```
# production.log

Process data files (123.45ms)
```

Fighting against slow queries

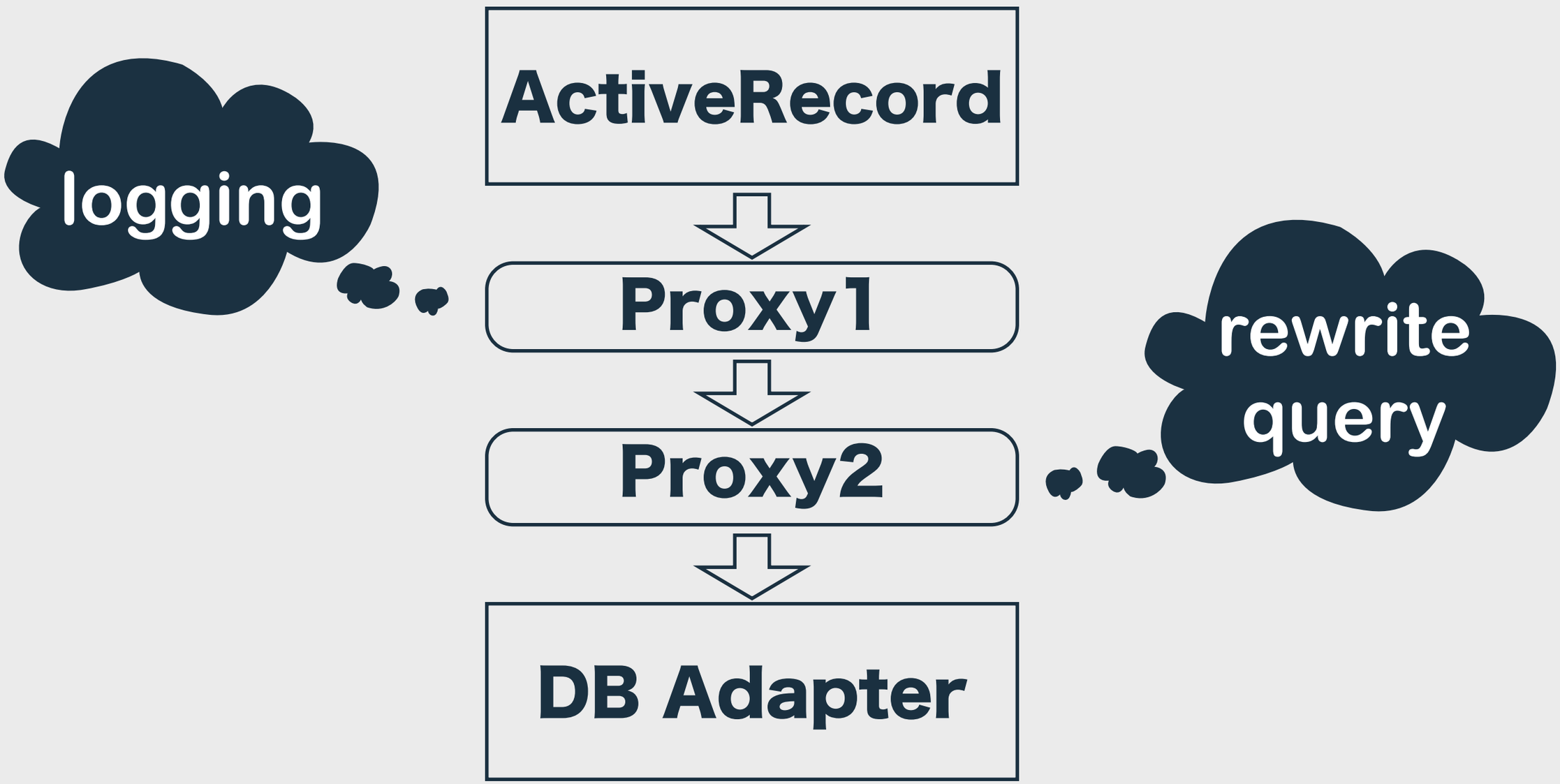
Arproxy

https://github.com/cookpad/arproxy



Arproxy

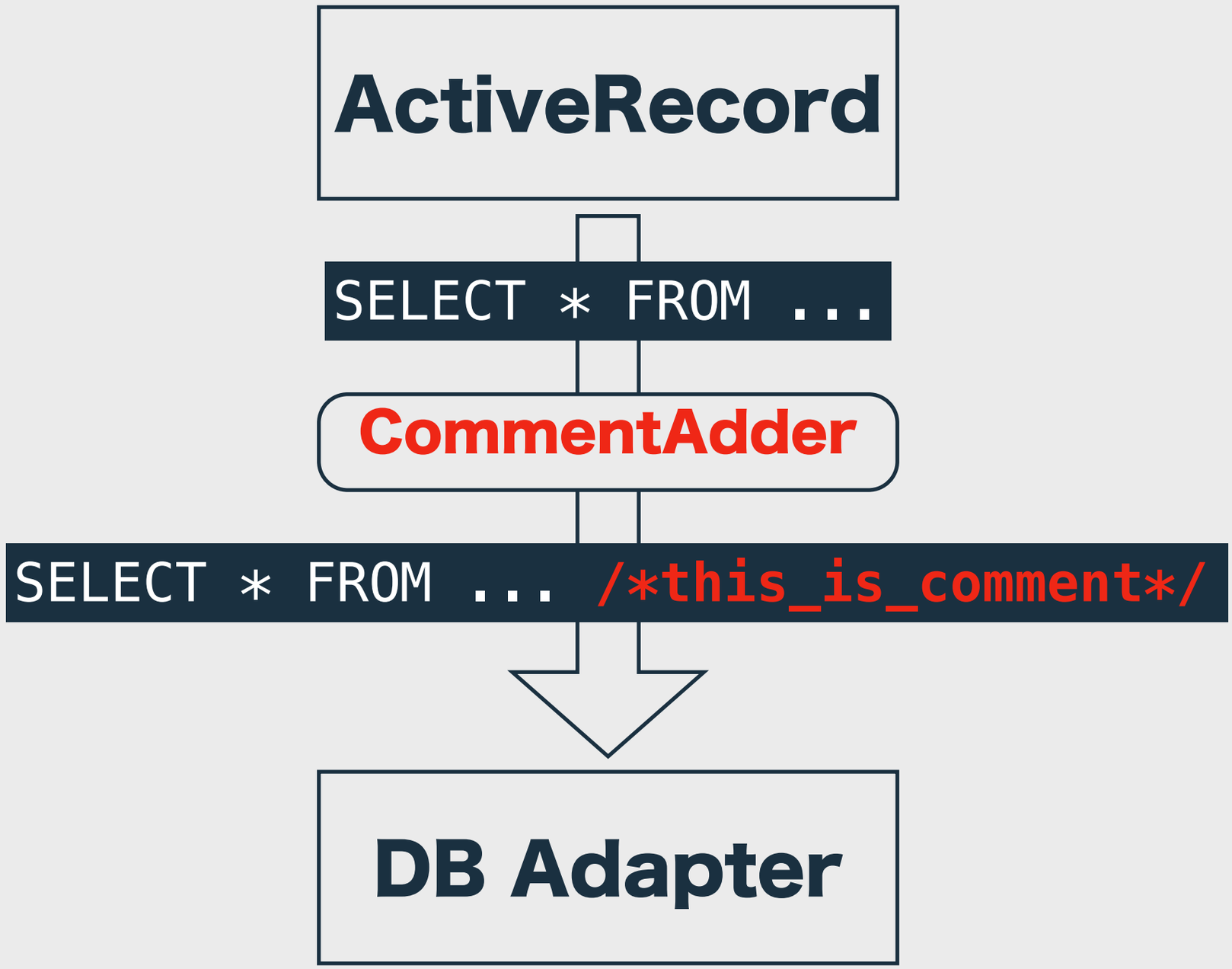
https://github.com/cookpad/arproxy



Arproxy example

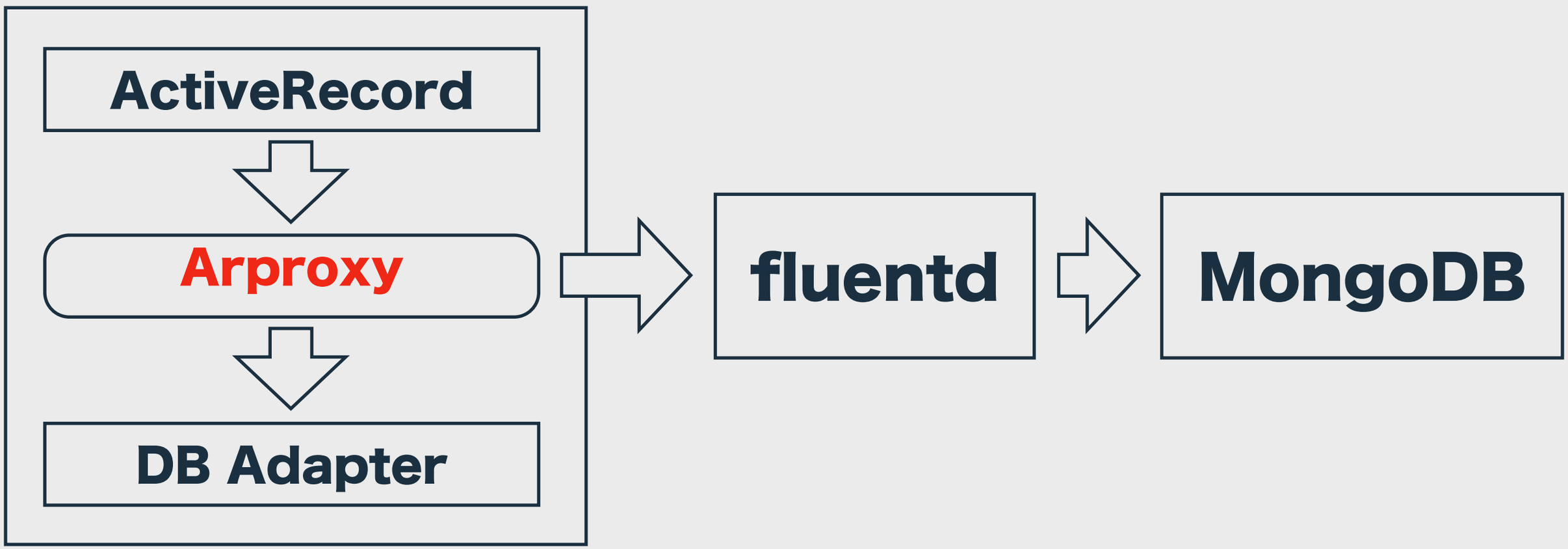
```ruby
# config/initializers/arproxy.rb
class CommentAdder < Arproxy::Base
  def execute(sql, name=nil)
    sql += " /*this_is_comment*/"
    super(sql, name)
  end
end

Arproxy.configure do |config|
  config.use CommentAdder
end
```

Arproxy example



Logging slow queries using Arproxy

COOKPAD



InfraMonitor Home Errors SlowQueries

COOKPAD

SlowQueries

SQL SELECT.*LIMIT 5000000 (Regular Expression)

MATCH: app_host db_host database username

NOT: app_host db_host database username

ORDER BY logged at execution time desc asc

search

2013-05-29 00:46:27 +0900 (2260ms)

app_host

db_host

database

username

```
SELECT COUNT(*) FROM `first_tsukurepos` WHERE `first_tsukurepos`.`recipe_id` =
```

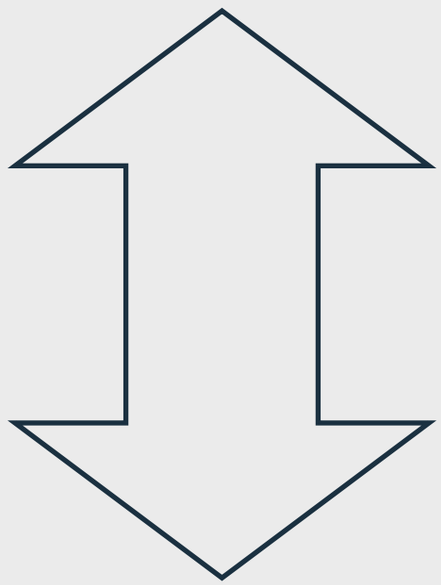
2013-05-29 00:45:30 +0900 (2060ms)

app_host

Conclusion

Trade-off in performance tuning

Developers



Users, Servers

PR

We're hiring

[en, ja] http://info.cookpad.com/jobs/

PR 2

Time to hack, guys

24-hour hack-a-thon

June 15 - 16

[ja] http://info.cookpad.com/24contest4

[en] http://info.cookpad.com/24contest4_en

Thank you for listening