Rails Conf 2024 DETROIT

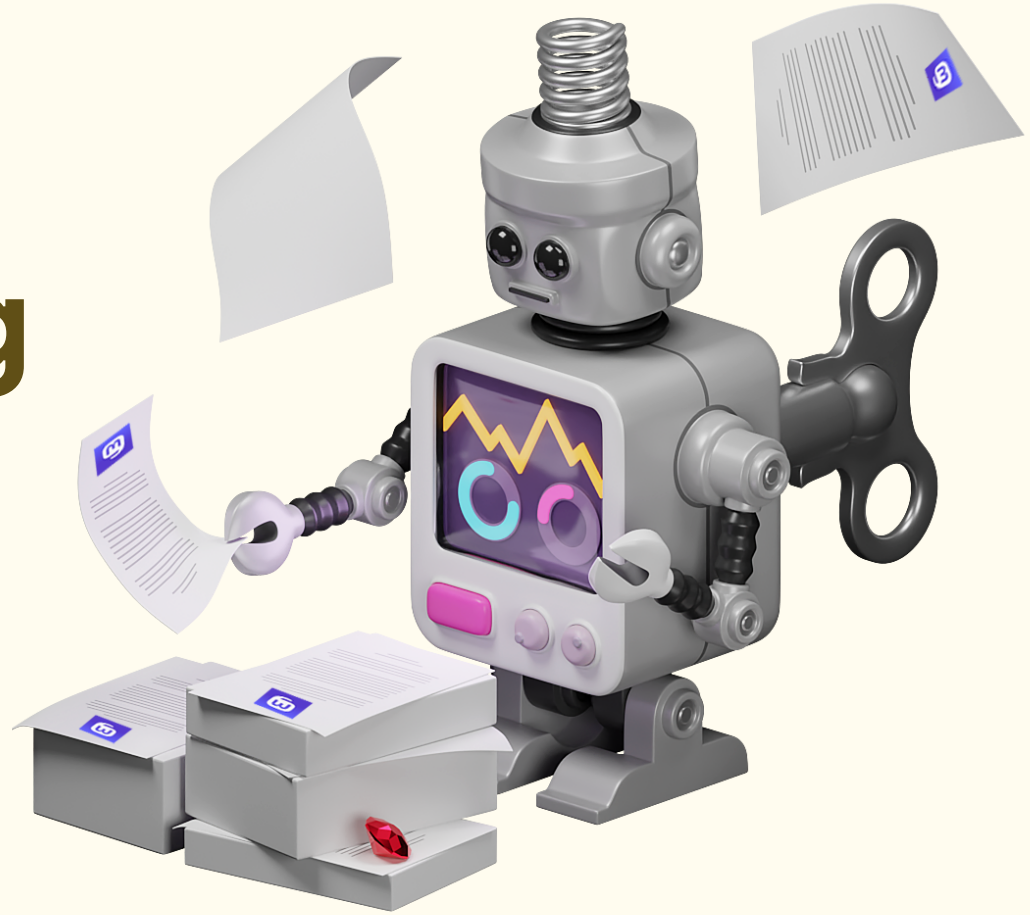
FROM SLOW TO GO: RAILS TEST PROFILING HANDS-ON

VLADIMIR DEMENTYEV

Principal Engineer

Evil Martians

From slow to go: Rails test profiling hands-on



Vladimir Dementyev

Evil Martians

Warming up

Or waiting for everyone's `dip rspec` to pass

Why are we here?

What's the plan?

Who are you?

You are here ==
your tests are slow

Slow tests

Longer CI builds (also more expensive)

Longer local runs (or "local CI" builds)

Slow tests

Longer CI builds (also more expensive)

Longer local runs (or "local CI" builds)

Longer feedback loops

More distraction, less productivity

From slow to go

🔍 **Identify** slowness causes and patterns

💊 **Treat** them well

🔁 **Repeat** until satisfied

Steven R. Baker
@srbaker

Not sure if it's relevant, but I notice a lot of people focus on "making the slow tests faster." On anything but the slowest of test suites, you get more mileage out of making often-used things faster.

The slowest tests are often slower for a difficult reason.

TestProf

test-prof.evilmartians.io

Specialized profiler for Ruby test suites

Optimization toolbox to speed up tests with less refactoring

TestProf

test-prof.evilmartians.io

Used by GitHub, Discourse, Gitlab, Dev.to, and many more

Used by YOU at this workshop!

github.com/palkan

Vladimir Dementyev

palkan

Code & Cats & Rock'n'Roll!

Edit profile

Sponsors dashboard

1.4k followers · **6** following

@evilmartians

Seattle

palkan / README.md

Hey / Привет 👋

My name is Vladimir (or *Вова*), and I'm a principal backend engineer at @evilmartians.

> 📕 My book "Layered design for Ruby on Rails applications" is avalable now: Amazon | Packt

I'm currently working on:

- Making real-time Ruby apps rock with @anycable (built with Ruby and Go).
- Bringing edge Ruby features to all Rubies with @ruby-next—a transpiler for Ruby.
- Helping Ruby devs to write faster tests with test-prof.
- ...and other projects, such as: Action Policy, Logidze, Anyway Config and many more.

Pinned

anycable/anycable Public

Polyglot replacement for Ruby WebSocket servers with Action Cable protocol

Ruby ☆ 1.8k 88

ruby-next/ruby-next Public

Ruby Next makes modern Ruby code run alternative implementations

Ruby ☆ 674 40

test-prof/test-prof Public

Ruby Tests Profiling Toolbox

Ruby ☆ 1.7k 140

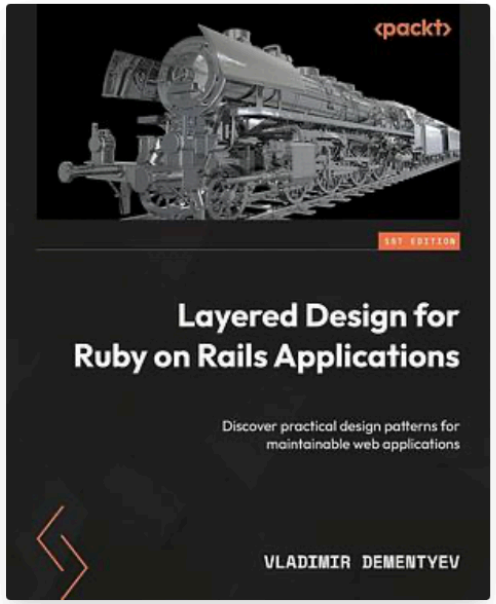
action_policy Public

Authorization framework for Ruby/Rails a

Ruby ☆ 1.2k 85

evilmartians.com

Stats Reading Challenges Community Giveaways Search all books...



Layered Design for Ruby on Rails Applications: Discover practical design patter for maintainable web applic

Vladim

298 pa

nonf

andycroll's review
Go to review page
informative medium-paced
4.0
Solid technical read.

Ruby on Rails is a
web applications from scra
productivity, leveraging the power of the
over-configuration principle, and the well-defined
view-controller pattern, assisting...

READ MORE

X

Disclaimer: Mastodon team already did some test performance work (via TestProf), so I had to revert a few optimizations in the workshop's version of the codebase (so we can have more fun)

The plan

General profiling (Stackprof, Vernier, sampling)

Focused profiling (TagProf, EventProf)

Factories overhead elimination (`let_it_be`)

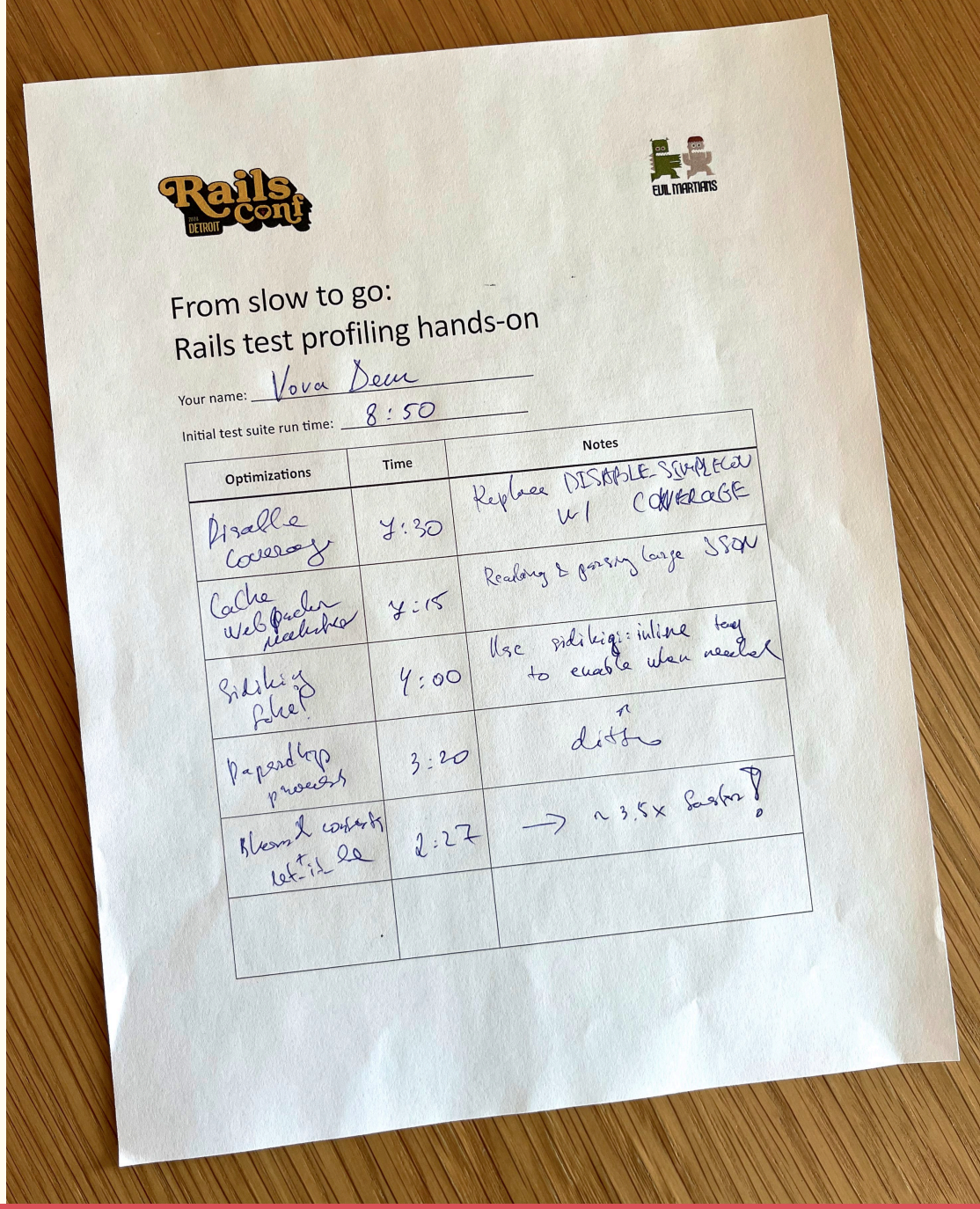
Results

Local time:

8:30 ➡ 2:30

CI time:

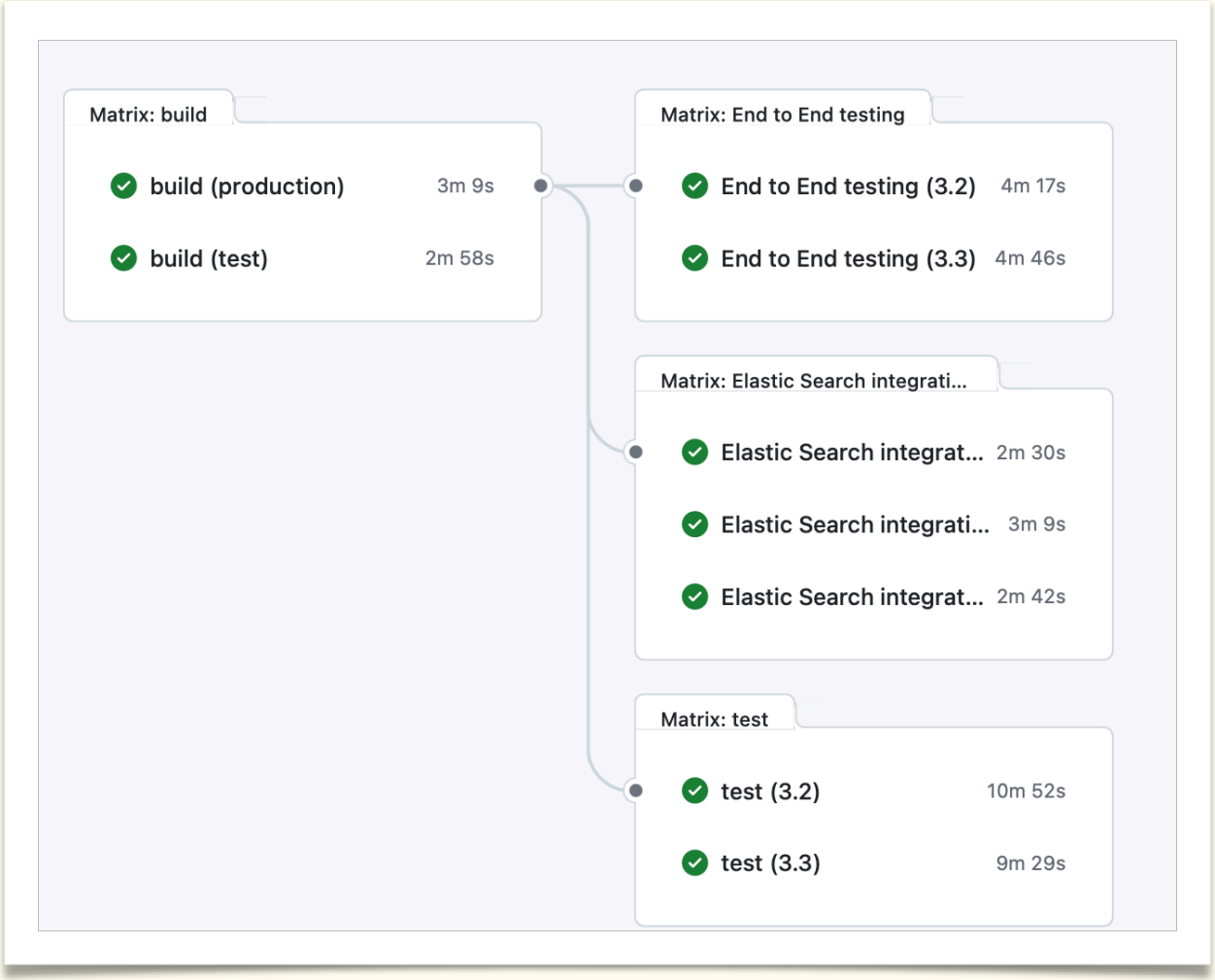
14:30 ➡ 8:00



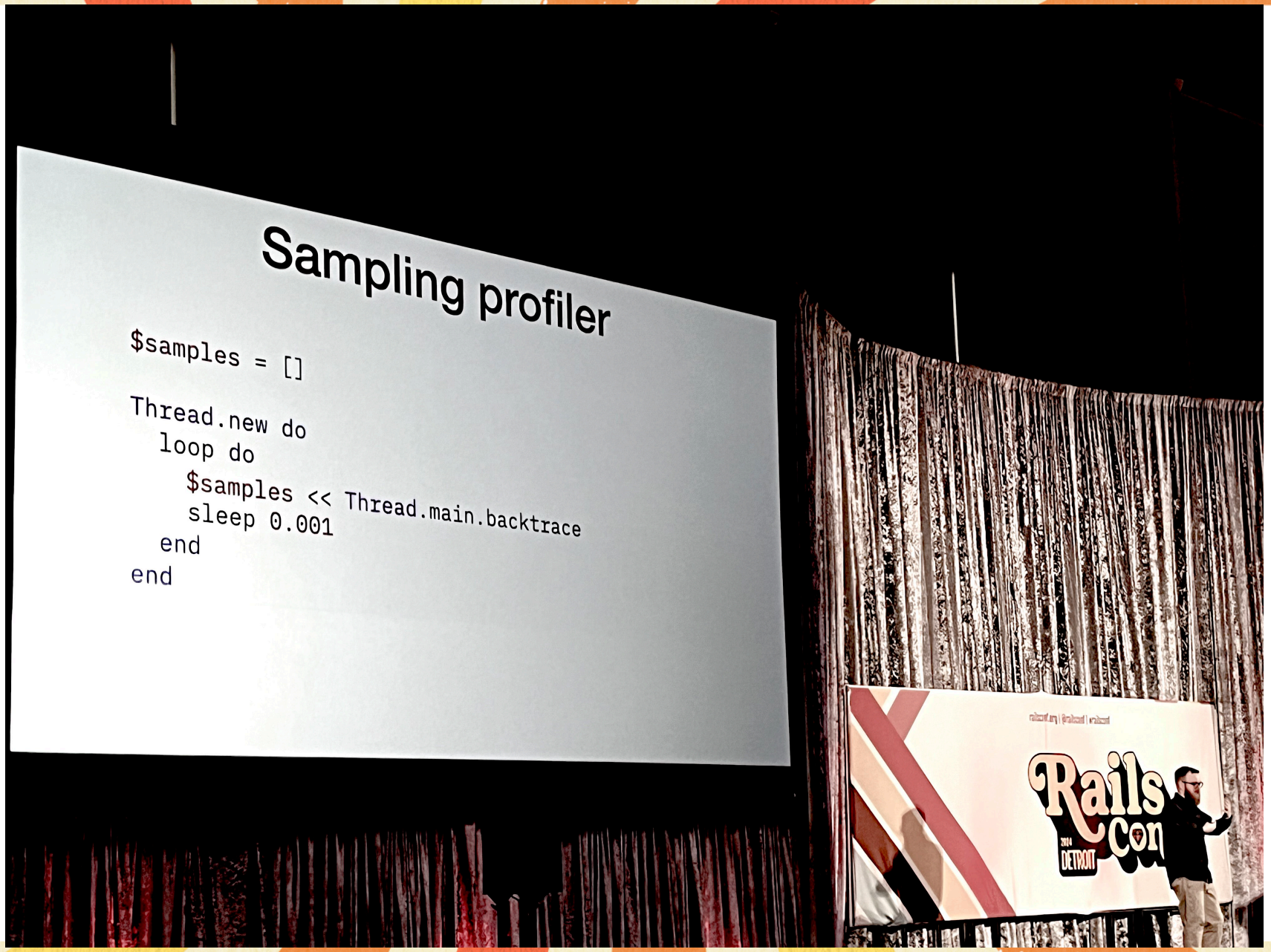
CI vs Rails tests

Rails tests occupy only for a portion of a CI build time

No need to optimize beyond other components



General profiling


Sampling profiler
$samples = []
Thread.new do
loop do
$samples << Thread.main.backtrace
sleep 0.001
end
end
Rails
Con
DETROIT

```ruby
# rails_helper.rb

TestProf::EventProf.monitor(
  Paperclip::Attachment,
  'paperclip.post_process',
  :post_process
)
```

Workshop instructions

```
TAG_PROF=type \
TAG_PROF_FORMAT=html \
TAG_PROF_EVENT=sql.active_record,factory.create,
sidekiq.inline,paperclip.post_process \
be rspec
```

Findings

Coverage: `opt-out` ➡ `opt-in`

```
gem "debug", require: false
```

Webpacker: `cache_manifest: true`

Findings

Logging—no effect when disk write is fast (SSD, Docker volumes on Mac)

Thousands of I18n YAMLs—handled by Bootsnap

Rails profiling story, or how I caught Faker trying to teach my app Australian Slang

February 12, 2019

Topics

Backend Ruby on Rails Ruby

Share this post on

Twitter Facebook LinkedIn

Vladimir Dementyev
Principal Backend Engineer

General profiling

Do not require test-specific profiling skills

Takes the most significant amount of time (sampling helps)

Highlights low-hanging fruits

Focused profiling

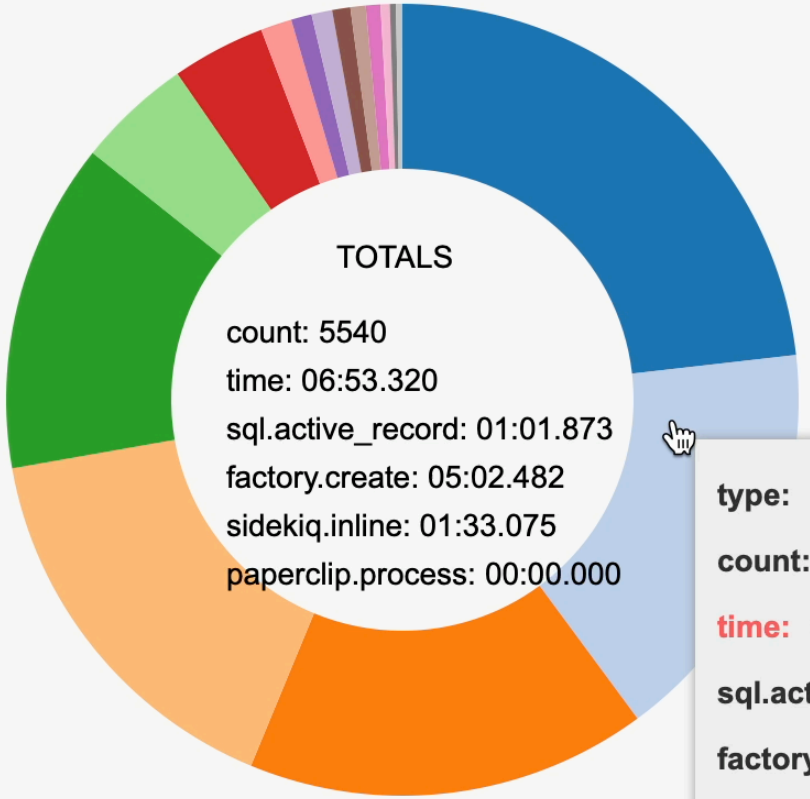
Focused profiling

Similar tests has similar problems

Focus on most time consuming test types, not individual outliers

TagProf Report

- (•) time
- ( ) sql.active_record
- ( ) factory.create
- ( ) sidekiq.inline
- ( ) paperclip.process



- request
- model
- service
- controller
- lib
- worker
- cli
- fabrication
- feature

type: model

count: 1115 (20.1%)

time: 01:09.003 (16.7%)

sql.active_record: 00:09.281 (15%)

factory.create: 01:00.863 (20.1%)

sidekiq.inline: 00:15.082 (16.2%)

Focused profiling

Reduced the number of tests to analyze by ~2x

Divided tests into smaller groups (~1min run time) for further analysis

Tools

TagProf

EventProf

RspecStamp (RSTAMP)

Findings

Sidekiq: `inline!` ➡ `fake!`

`Paperclip::Testing.fake!`

TestProf: a good doctor for slow Ruby tests

August 15, 2017

Topics

Backend | Performance Audit and Optimization | Ruby | Ruby on Rails | Continuous Integration

Vladimir Dementyev
Principal Backend Engineer

Workshop instructions

```
# .rspec-local

--tag ~paperclip:process
```

What's next?

type	time	factory.create	total	%total	%time
request	01:35.555	01:02.509	1205	21.75	23.43
model	01:07.256	00:59.407	1115	20.13	16.49
service	01:06.297	00:43.816	587	10.60	16.26
controller	01:06.291	00:45.765	833	15.04	16.25
lib	00:54.607	00:43.887	711	12.83	13.39
worker	00:19.236	00:12.761	234	4.22	4.72
cli	00:15.204	00:09.747	242	4.37	3.73

Factories

Workshop instructions

```
FPROF=1 \
EVENT_PROF=factory.create \
be rspec spec/models
```

Factories vs. slow tests

Factory cascades

Too many disposable records (no reuse or recycle)

Create vs. build

TestProf II: Factory therapy for your Ruby tests

May 25, 2018

Topics

Backend | Performance Audit and Optimization | Ruby | Ruby on Rails | Continuous Integration

Vladimir Dementyev
Principal Backend Engineer

Tools

FactoryProf

RSpecDissect

`before_all` / `let_it_be`

Findings

No cascades found

Many tests share the same context

Reusing records between examples brings up to 6x speed up!

Findings

Replacing let/let! with let_it_be works in ~90% of tests

The rest may require some refactoring (usually minor)

```
      context 'when status is private' do
-       let(:status) { Fabricate(:status, account: account, visibility: :private) }
+       before { status.update!(visibility: :private) }
```

Findings

Every file must be treated individually (routine)

Extracting shared contexts may help simplify future optimizations (e.g., adding fixtures)

Homework

Homework

Discover more TestProf tools (Autopilot, RuboCop cops, Factory Doctor)

Make YOUR tests faster

Become a Mastodon contributor!

Thank you

Visit TestProf and
Evil Martians at the Hack Day!