How to use flamegraphs to find performance problems

About me

Structure for the first half

- Background
- How to find slow pages
- How to investigate why they are slow

Why we care about performance

- Better user experience
- Customer-facing pages: affects conversion rate
- Easiest way to measure this is to measure the drop-out rate in your sales funnel.

Server costs

- Slow code in heavily used areas of a site = higher server costs.

@_jadedickinson

Scaling

- Makes scaling under peak load more difficult

Why do we need to profile apps?

@_jadedickinson

Why do we need to profile apps?

- Realistic view of what's causing slowness
- Intuition can fail

Why do we need to profile apps?

- Realistic view of what's causing slowness
- Intuition can fail
- Unlikely to catch slowness in code review

Profilers

- What is a profiler

Profilers

- What is a profiler
- Common Ruby profilers

How to find performance problems

- Start by prioritising, then:

How to find performance problems

- Start by prioritising, then:
- Follow an elimination process to find cause of slow code

How to find performance problems

- Start by prioritising, then:
- Follow an elimination process to find cause of slow code
- Measure results of any code change

@_jadedickinson

How to find performance problems

- Start by prioritising, then:
- Follow an elimination process to find cause of slow code
- Measure results of any code change
- Raise a PR and test in production

@_jadedickinson

Prioritising areas to speed up

- What pages are both heavily used and slow

Prioritising areas to speed up

- What pages are both heavily used and slow
- Of these, what would be the impact of speeding these up

Prioritising areas to speed up

- What pages are both heavily used and slow
- Of these, what would be the impact of speeding these up
- Context of the business you're in
- Homepage / Conversion funnel / Landing pages / Payments

@_jadedickinson

Prioritising areas to speed up

- What pages are both heavily used and slow
- Of these, what would be the impact of speeding these up
- Context of the business you're in
- Homepage / Conversion funnel / Landing pages / Payments
- Customer base

@_jadedickinson

Put together an ordered list

- Look at real results for real users

@_jadedickinson

Examples

Type Web

Most time consuming

AppraisalsController#edit	11.8%
UsersController#dashboard	9.35%
OnlineJobPlansController#calendar	7.2%
OnlineJobPlansController#show	4.25%
Admin::UsersController#index	3.25%
EdocsController#create	3.04%
JustificationsController#update	2.55%
ArtifactsController#index	2.31%
Admin::UsersController#search	2.31%
Devise::SessionsController#new	2.22%
OnlineJobPlansController#activity_summary	2.16%
AppraisalsController#pdf	1.7%
OnlineJobPlansController#pdf	1.62%
EleavesController#edit	1.61%
EdocsController#download	1.6%
EdocsController#index	1.38%
JobContentsController#index	1.31%
AppraisalDatasetsController#index	1.31%
Admin...isalMonthComplianceController#index	1.05%
Admin::EdocsController#index	0.99%

Type Web

Most time consuming

StockCarsController#show_listing	10.7%
ShowroomController#show	10.7%
Stock...ndationsController#index	5.71%
Remot...gurationsController#show	5.12%
StockCarsController#filter_search	4.71%
LeaseDealsController#index	3.49%
Showr...ow_matching_alternatives	3.24%
Api::...figurationsController#show	3.11%
CarCo...onsController#choose_trim	2.97%
Api::...ationsController#with_price	2.78%
StockCarsController#show	2.76%
Offer...r#remote_show_promotions	2.62%
Api::...derQuotesController#create	2.16%
CarCo...nsController#choose_filters	2.04%
Api::...rderQuotesController#show	1.87%
ShowroomController#index	1.83%
Api::...mmendationsController#top	1.75%
OffersController#show	1.73%
CarCo...Controller#choose_options	1.69%
Api::V2::EnquiriesController#index	1.65%

TIME PICKER
7 days ending 04/03, 9:58

SERVERS
All servers

Type Web

Most time consuming

ChopinJourneyController#handle	43.6%
Backoffice::ChopinJourneyController#handle	13.5%
Backoffice::RfqsController#show	13.3%
Backoffice::PolicySegmentsController#show	5.35%
PingController#lb_health_check	2.92%
Ci5::YourDocumentsController#show	2.28%
Backoffice::NotesController#create	1.73%
SearchController#index	1.29%
Backoffice::LinkedRfqsAndPoliciesController#index	1.27%
v6: /policies/:id/choco_quote (POST)	1.01%

@_jadedickinson

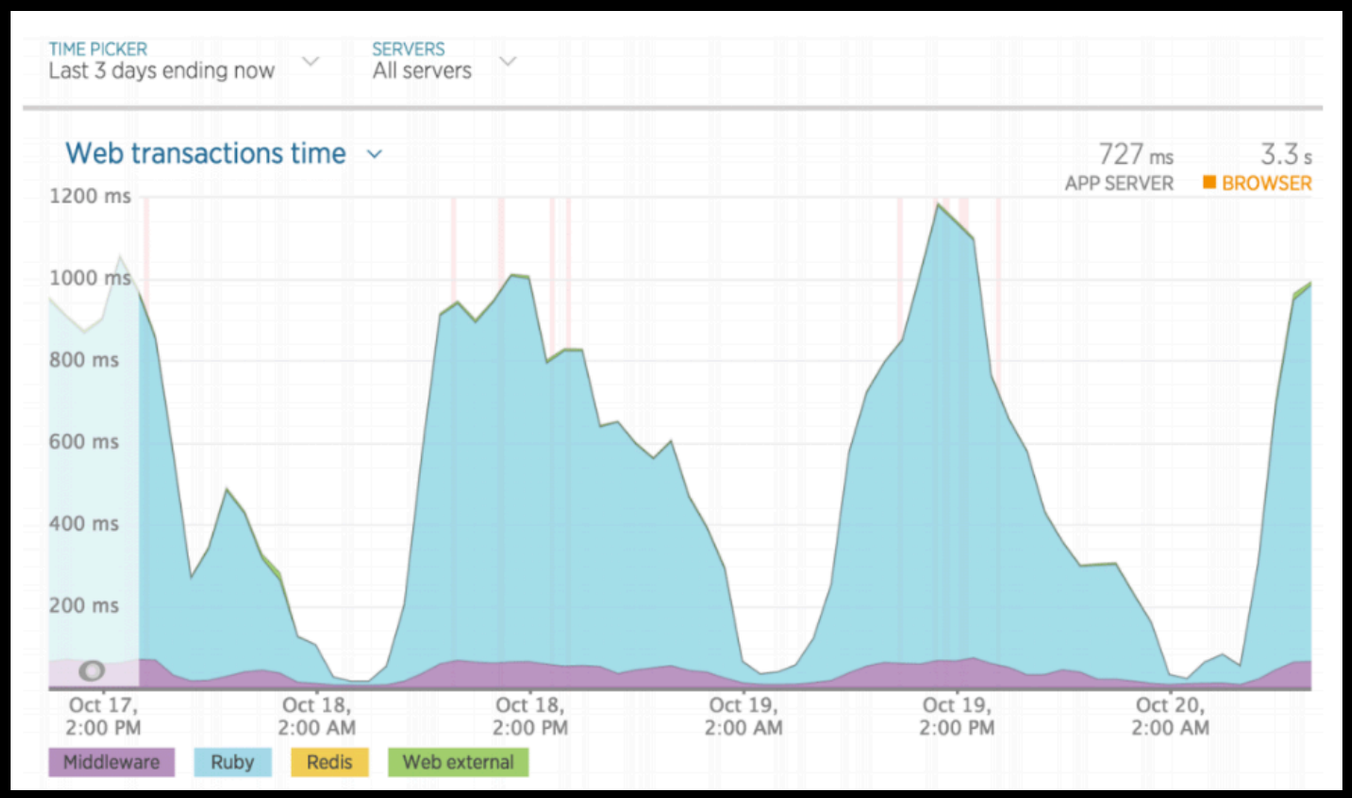
Pick your starting point

- Highest impact on product
- Consider architecture where relevant

@_jadedickinson

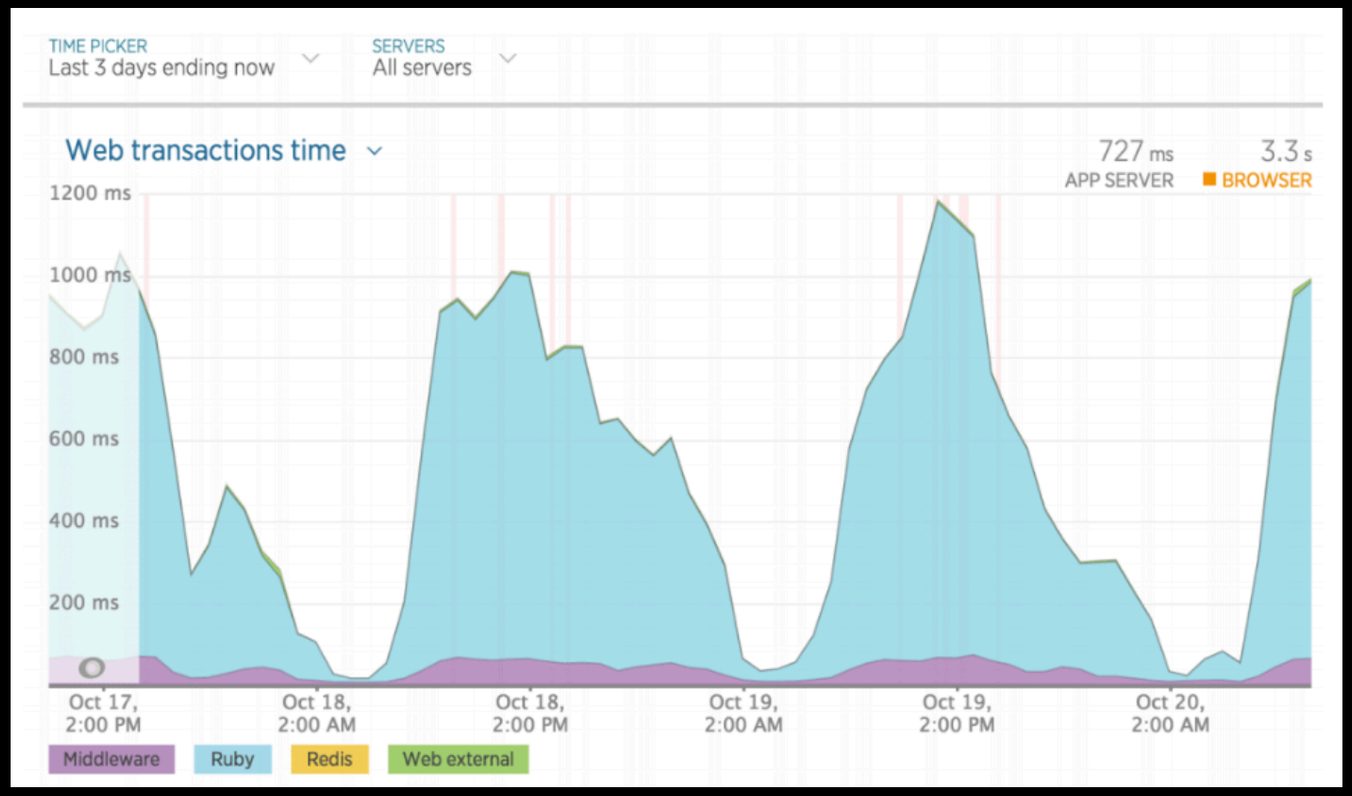
Elimination process

Elimination process



@_jadedickinson

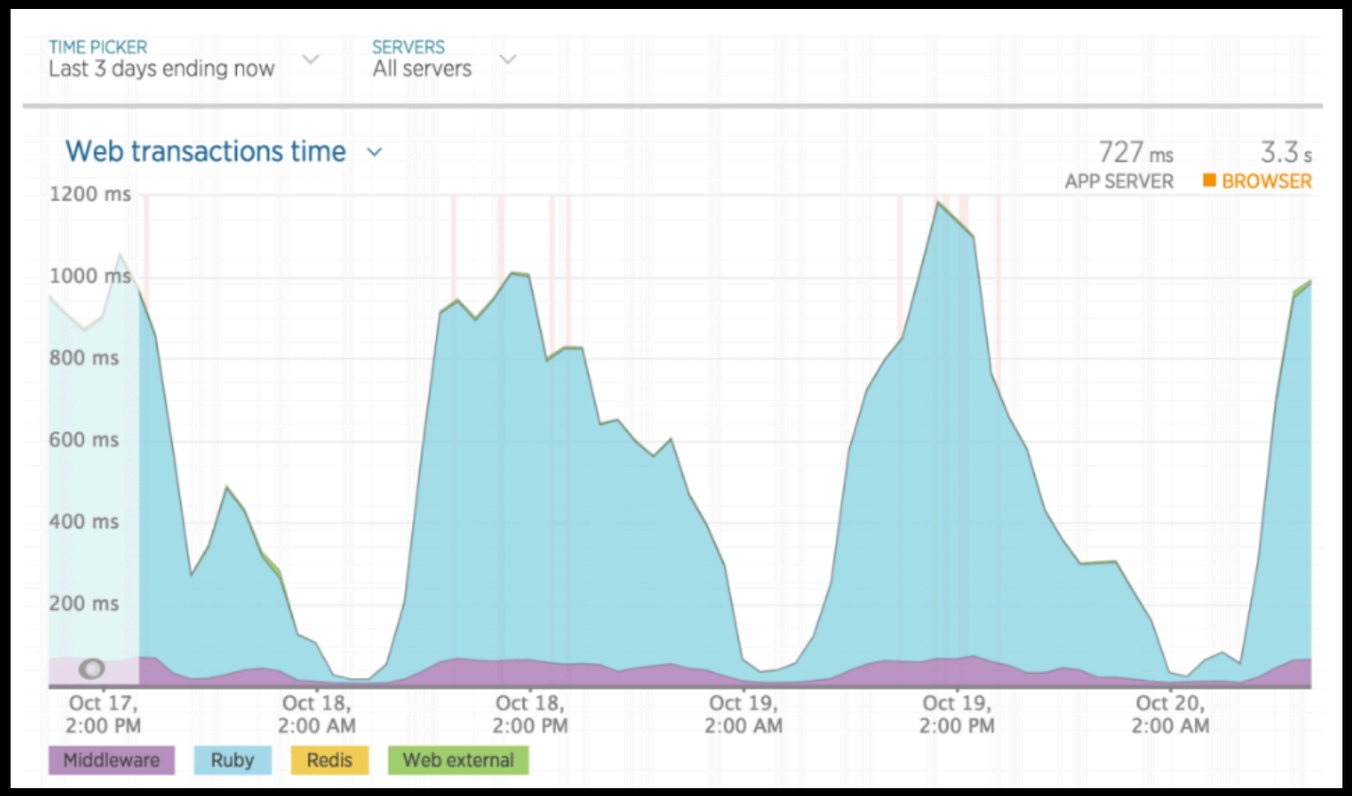
Elimination process



- Is it the frontend?

@_jadedickinson

Elimination process



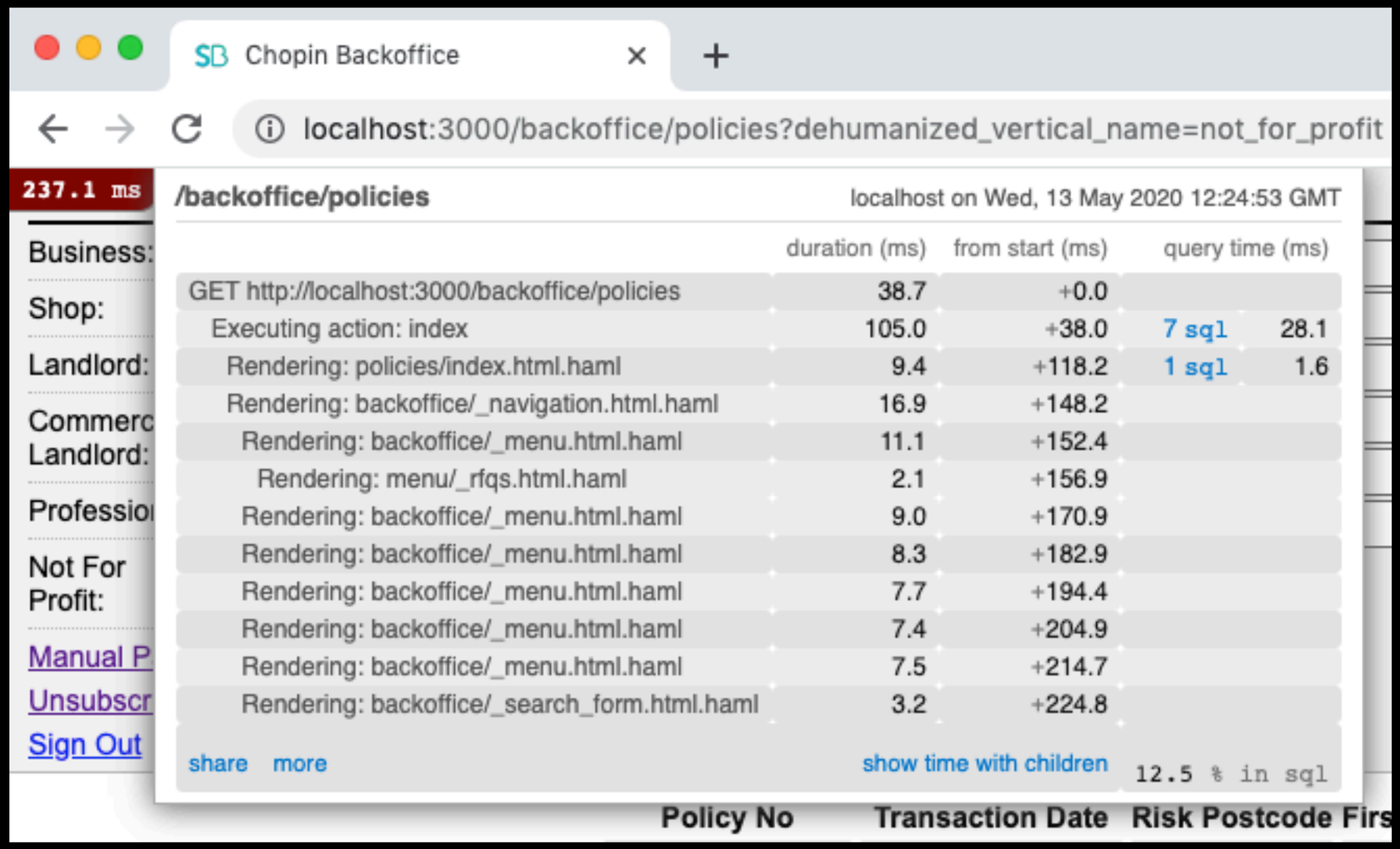
- Is it the frontend?
- Is it the backend?
- Where in the backend?

@_jadedickinson

EXERCISE 1

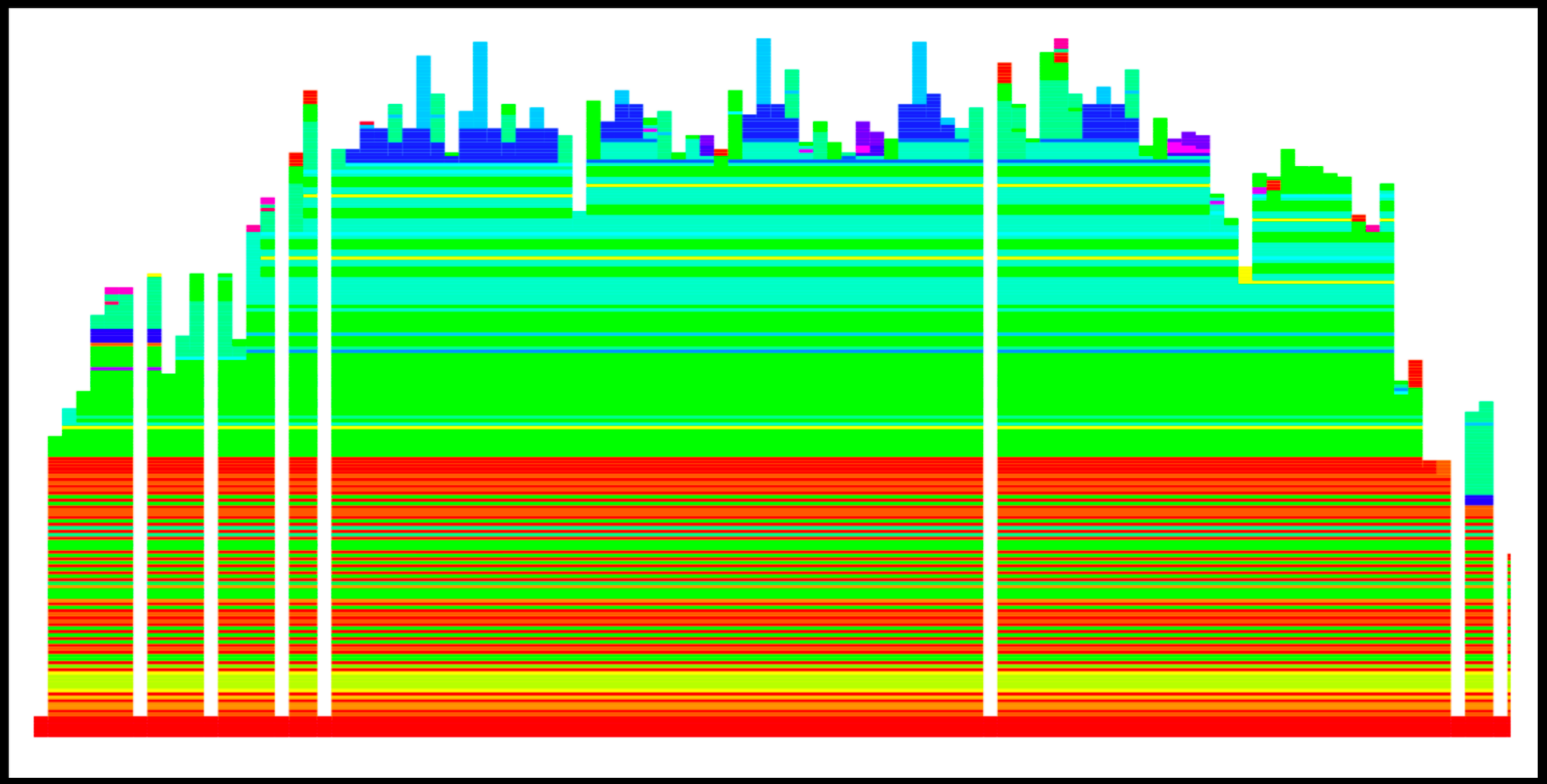
http://localhost:3000

Rack-mini-profiler



@_jadedickinson

The flamegraph



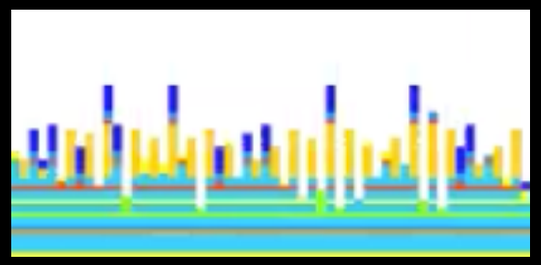
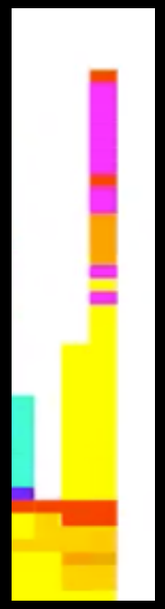
@_jadedickinson

EXERCISE 2

http://localhost:3000/cohorts?pp=flamegraph

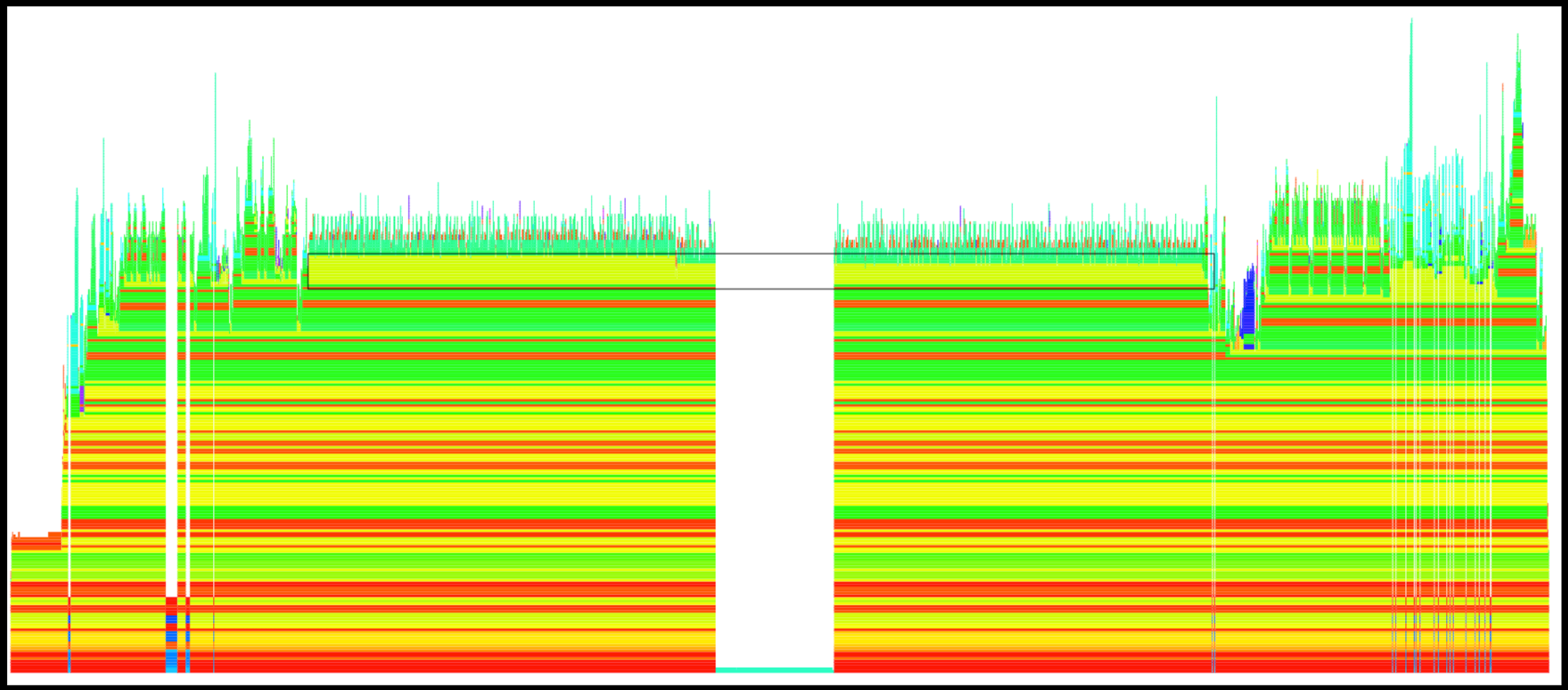
@_jadedickinson

Patterns to look for in a flamegraph

- Big horizontal section same colour
- Hedgehog spikes 
- Random peak in a different colour 
- Chunk cut out of the flamegraph

@_jadedickinson

Example



@_jadedickinson

EXERCISE 3

http://localhost:3000/enclosures?pp=flamegraph

@_jadedickinson

EXERCISE 4

http://localhost:3000/animals?pp=flamegraph

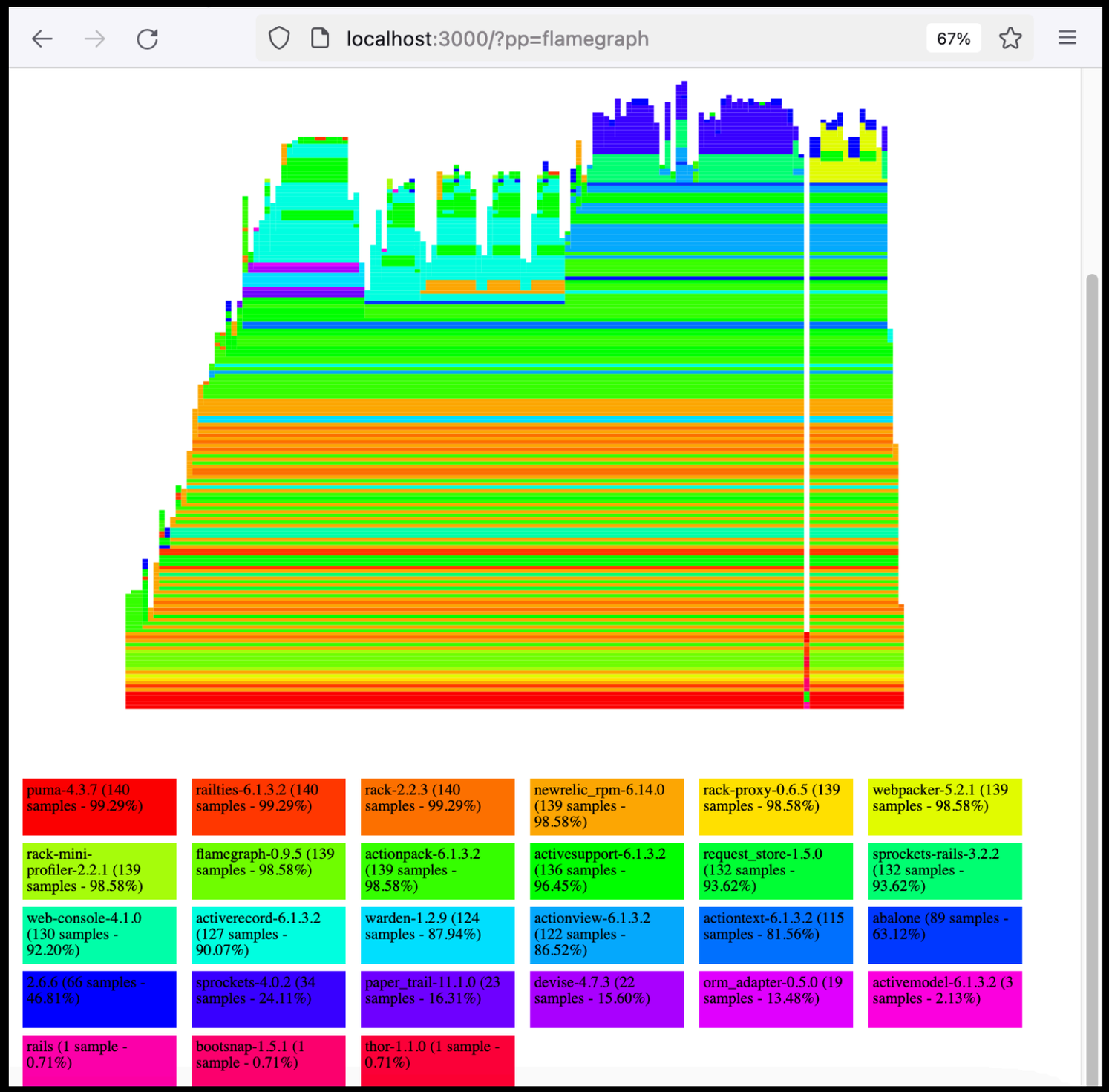
@_jadedickinson

EXERCISE 5

http://localhost:3000/?pp=flamegraph

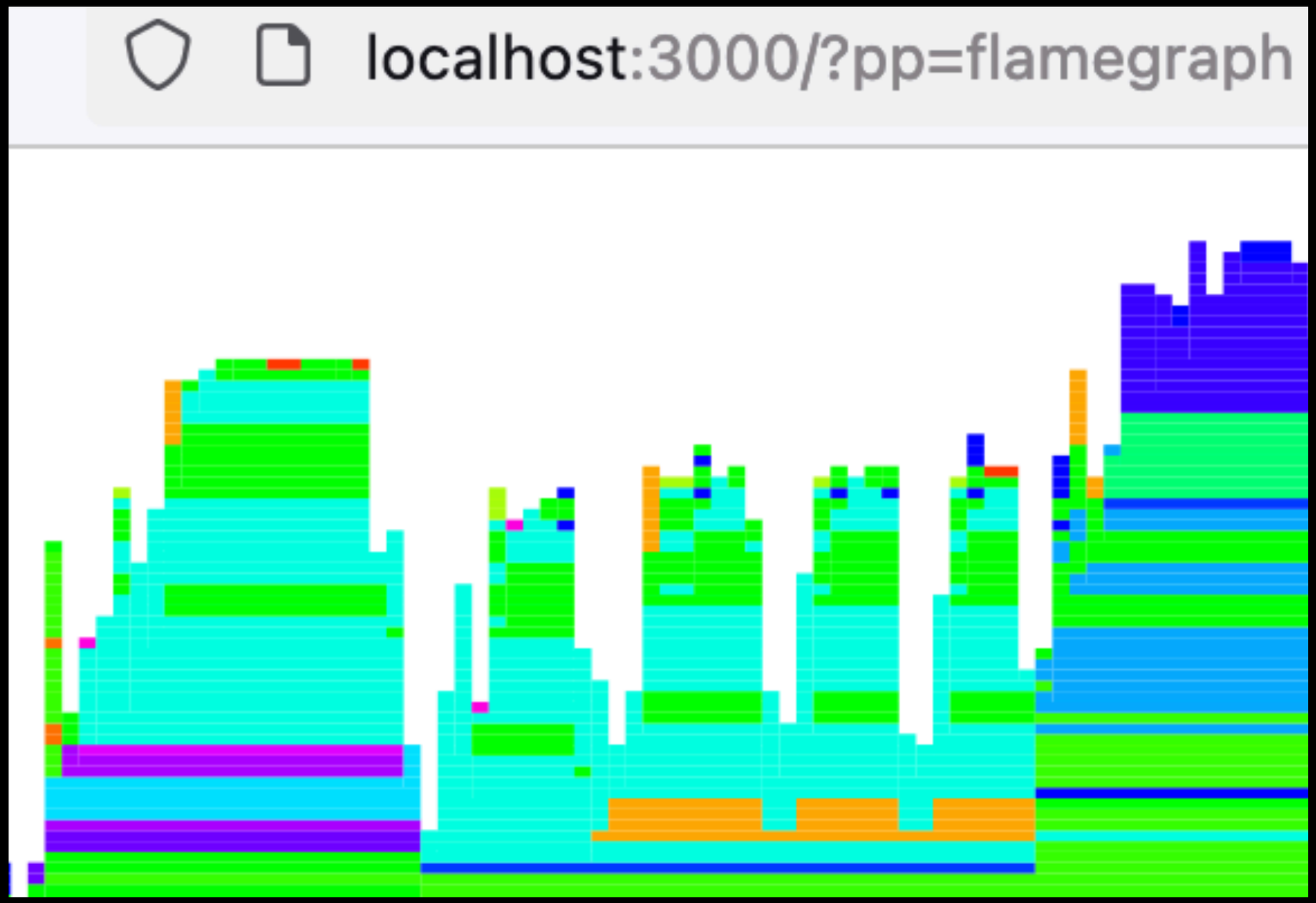
@_jadedickinson

EXERCISE 5



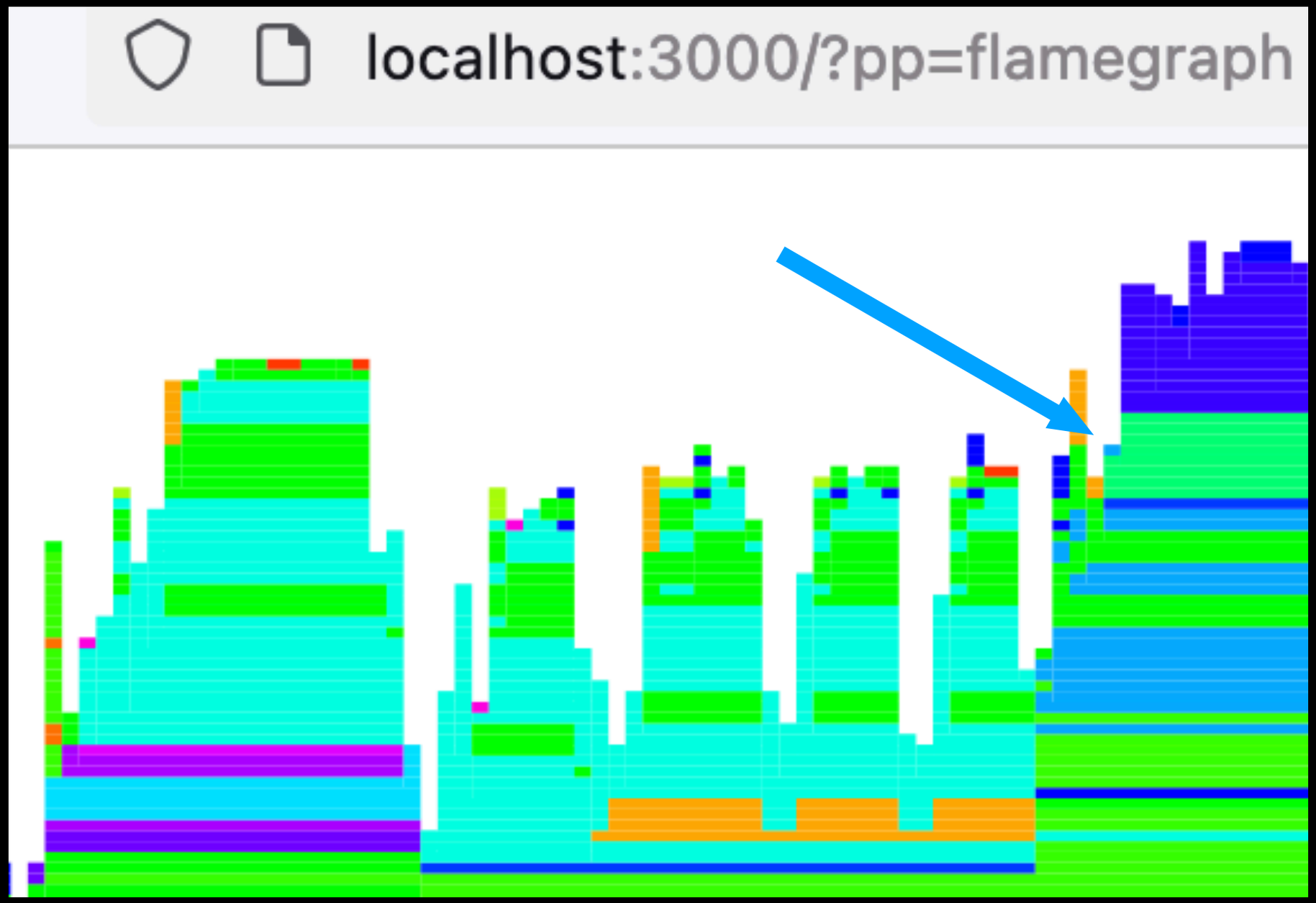
@_jadedickinson

EXERCISE 5



@_jadedickinson

EXERCISE 5



@_jadedickinson

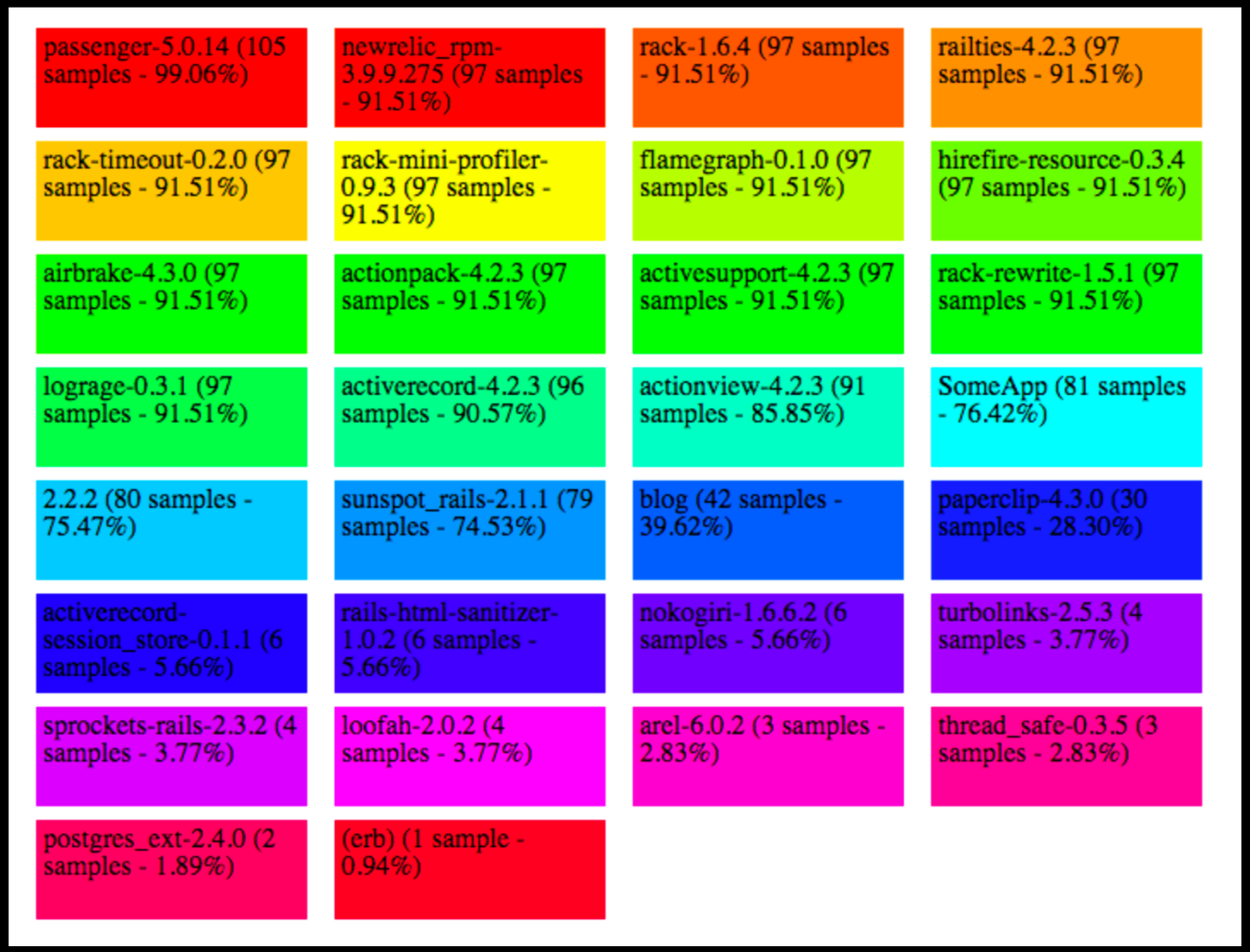
Seeing stack traces from an individual frame

Frame Info

/Users/Jade.Dickinson/projects/chopin/app/models/vertical.rb:15:in `Vertical#questionnaire'	(1048 samples - 34.30%)
/Users/Jade.Dickinson/projects/chopin/app/models/rfq.rb:1473:in `Rfq#questionnaire'	(1124 samples - 36.79%)
/Users/Jade.Dickinson/projects/chopin/app/helpers/backoffice/rfqs_helper.rb:28:in `Backoffice::RfqsHelper#select_sof_questions'	(1127 samples - 36.89%)
/Users/Jade.Dickinson/projects/chopin/app/helpers/backoffice/rfqs_helper.rb:20:in `Backoffice::RfqsHelper#sof_answers_in_questionnaire_order'	(504 samples - 16.50%)
/Users/Jade.Dickinson/projects/chopin/app/helpers/backoffice/rfqs_helper.rb:20:in `Backoffice::RfqsHelper#sof_answers_in_questionnaire_order'	(504 samples - 16.50%)
/Users/Jade.Dickinson/projects/chopin/app/presenters/backoffice/base_rfq_presenter.rb:226:in `Backoffice::BaseRfqPresenter#all_sof_answers'	(504 samples - 16.50%)
/Users/Jade.Dickinson/projects/chopin/app/presenters/backoffice/base_rfq_presenter.rb:78:in `Backoffice::BaseRfqPresenter#rfq_details'	(1130 samples - 36.99%)
/Users/Jade.Dickinson/projects/chopin/app/views/backoffice/rfqs/_rfq_details_table.html.haml::in `ActionView::CompiledTemplates#_app_views_backoffice_rfqs__rfq_details_table_html_haml__1517135409571939764_70110887633100'	(1143 samples - 37.41%)
… actionview-5.2.4.1/lib/action_view/template.rb:156:in `ActionView::Template#render'	(2034 samples - 66.58%)
… activesupport-5.2.4.1/lib/active_support/notifications.rb:166:in `#x00007f87f70a1370>.instrument'	(2080 samples - 68.09%)
… actionview-5.2.4.1/lib/action_view/template.rb:353:in `ActionView::Template#instrument_render_template'	(2034 samples - 66.58%)
… actionview-5.2.4.1/lib/action_view/template.rb:156:in `ActionView::Template#render'	(2034 samples - 66.58%)
… actionview-5.2.4.1/lib/action_view/renderer/partial_renderer.rb:332:in `ActionView::PartialRenderer#render_partial'	(1875 samples - 61.37%)
… actionview-5.2.4.1/lib/action_view/renderer/abstract_renderer.rb:40:in `ActionView::AbstractRenderer#instrument'	(1977 samples - 64.71%)
… activesupport-5.2.4.1/lib/active_support/notifications.rb:166:in `#x00007f87f70a1370>.instrument'	(2080 samples - 68.09%)
… activesupport-5.2.4.1/lib/active_support/notifications/instrumenter.rb:19:in `ActiveSupport::Notifications::Instrumenter#instrument'	(2080 samples - 68.09%)
… activesupport-5.2.4.1/lib/active_support/notifications.rb:166:in `#x00007f87f70a1370>.instrument'	(2080 samples - 68.09%)
… actionview-5.2.4.1/lib/action_view/renderer/abstract_renderer.rb:40:in `ActionView::AbstractRenderer#instrument'	(1977 samples - 64.71%)
… actionview-5.2.4.1/lib/action_view/renderer/partial_renderer.rb:332:in `ActionView::PartialRenderer#render_partial'	(1875 samples - 61.37%)
… actionview-5.2.4.1/lib/action_view/renderer/partial_renderer.rb:297:in `ActionView::PartialRenderer#render'	(1901 samples - 62.23%)

@_jadedickinson

The flamegraph – time spent



@_jadedickinson

Investigating further

app > models > rfq.rb

```
1511
1512    def questionnaire
1513      vertical.questionnaire(web_rfq.merge({ Site: site }))
1514    end
```

app > models > vertical.rb

```
14
15    def questionnaire(answers={}, to_merge={})
16      answer_set = answers.merge(to_merge)
17      site = { "site" => answer_set.fetch("site", "simplybusiness") }
```

@_jadedickinson

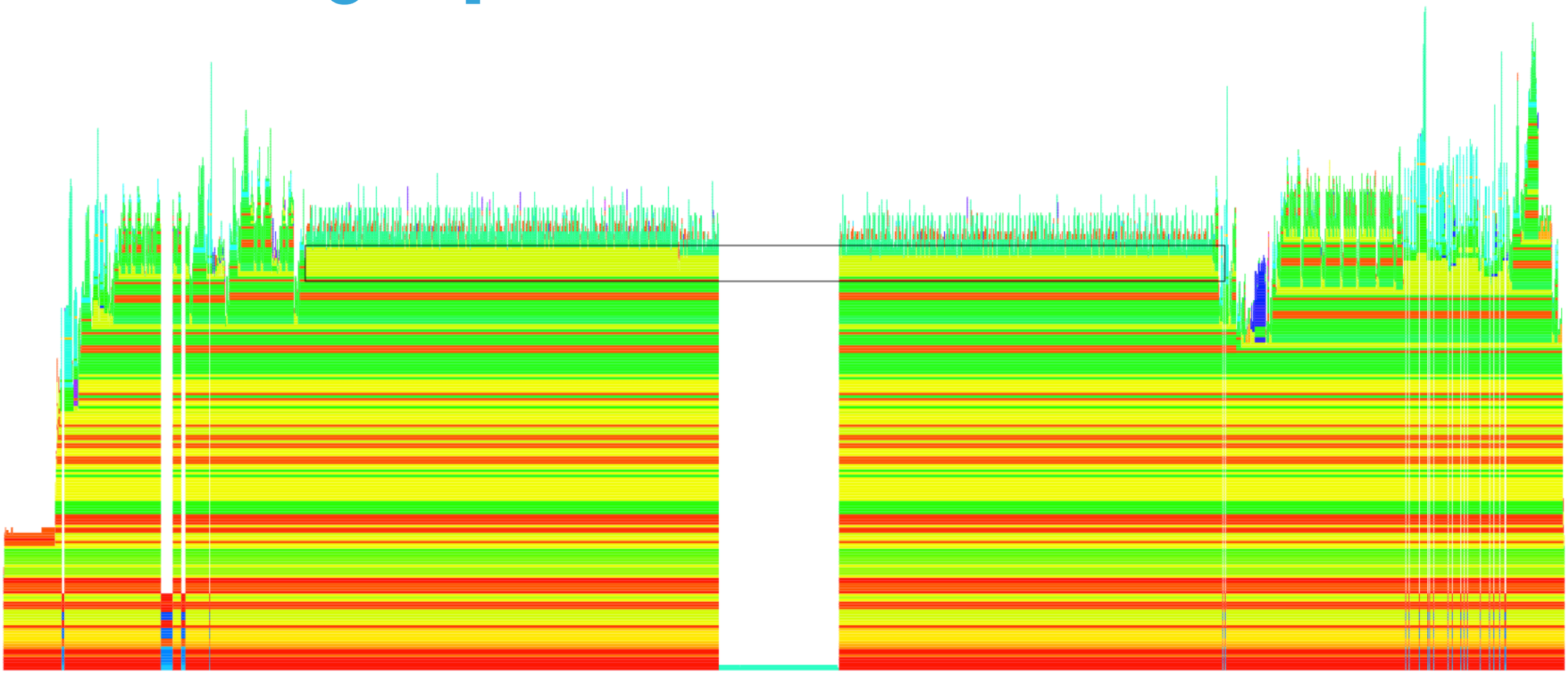
Avoiding these unnecessary calls

app > models > rfq.rb

```
1511
1512    def questionnaire
1513      @questionnaire ||= vertical.questionnaire(web_rfq.merge({ Site: site }))
1514    end
```

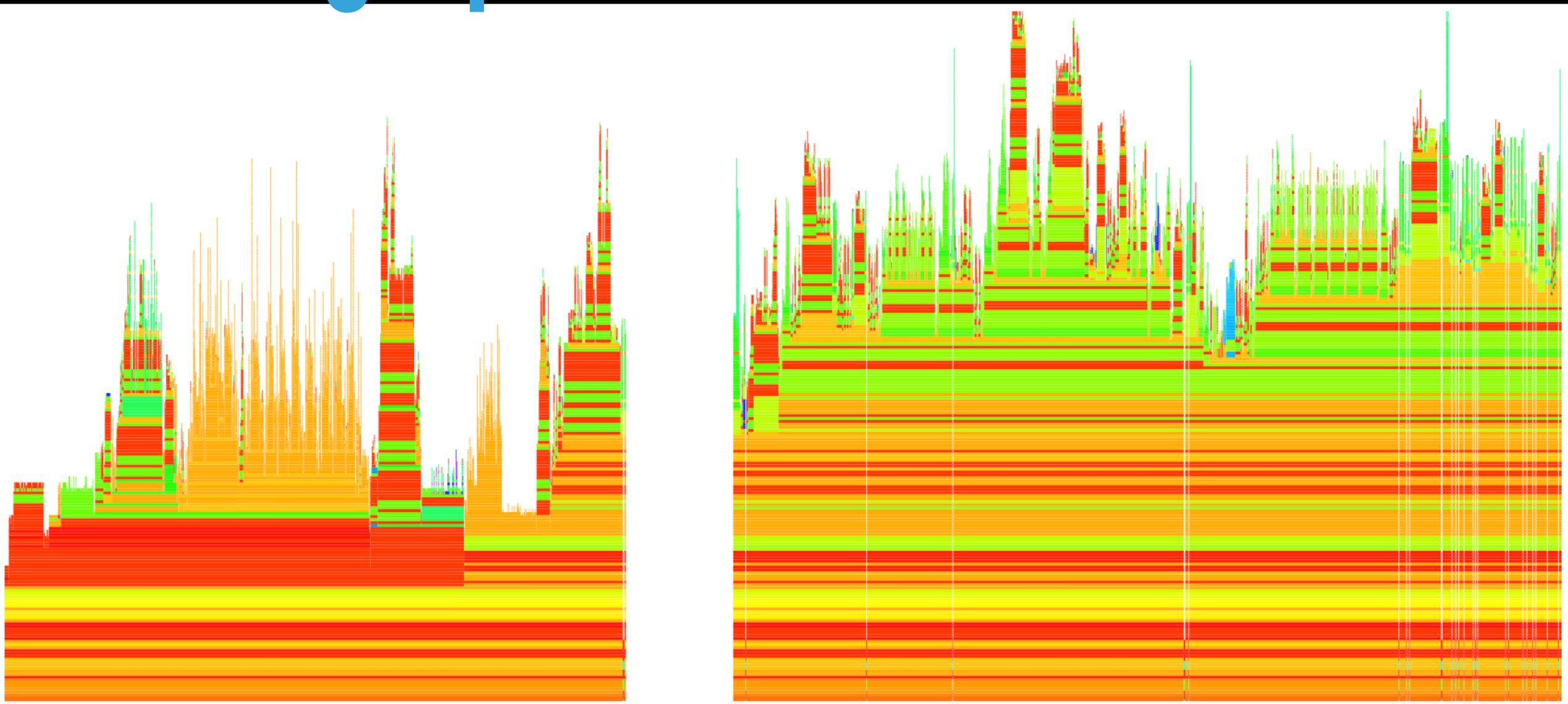
@_jadedickinson

Flamegraph before



@_jadedickinson

Flamegraph after



@_jadedickinson

Benchmarking results

Benchmarking results

ab -t 30 -H 'host: localhost' -H 'Cookie:' http://127.0.0.1:3000/

Before
Finished 29 requests
Time taken for tests: 30.627 seconds

Requests per second: 0.95 [#/sec] (mean)
Time per request: 1056.108 [ms] (mean)

After
Finished 51 requests
Time taken for tests: 30.573 seconds

Requests per second: 1.67 [#/sec] (mean)
Time per request: 599.467 [ms] (mean)

SECOND HALF

@_jadedickinson

Structure for the second half

- Adapting your approach
- Beyond getting your PR into prod
- Deep dive on flamegraphs

@_jadedickinson

Adapting to what stage your product is in

@_jadedickinson

Adapting to what stage your product is in

- Early stage
- Rapid growth stage
- Growth has stabilised

@_jadedickinson

Early stage

- Finding product-market fit
- Less likely to have slow pages
- Hold this advantage

@_jadedickinson

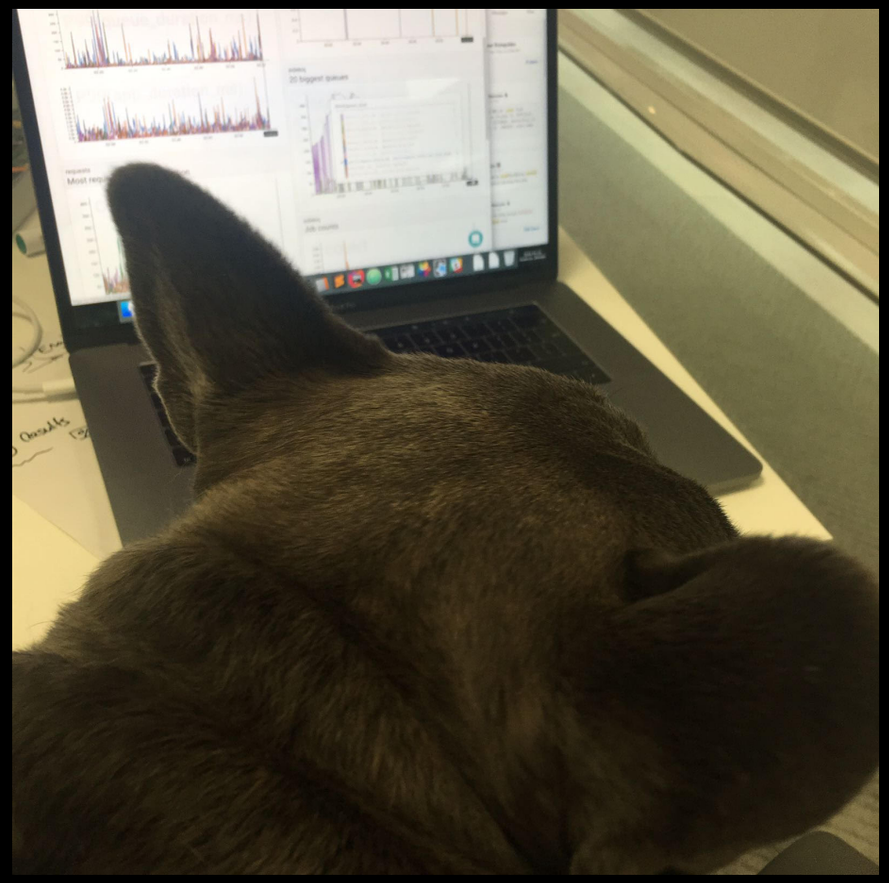
Rapid growth stage

- Losing the sale
- “The site is down”
- “It takes ages to load”
- Start thinking about site reliability

@_jadedickinson

Site reliability on one slide

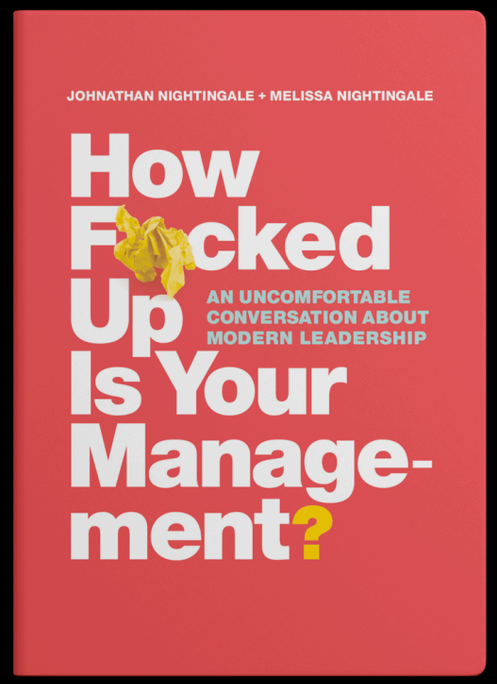
- Availability: keeping the site up
- Response times: keeping the site fast
- Proactive work: perf optimisation, plan for scale
- Reactive work e.g. major incident response
- Performance targets aka SLOs



@_jadedickinson

Growth has stabilised

- Research the competition
- Drive growth through optimisation
- Bring down your server costs



@_jadedickinson

Pushing on

@_jadedickinson

Pushing on

- Marginal gains

Pushing on

- Marginal gains
- Bringing your team along with you

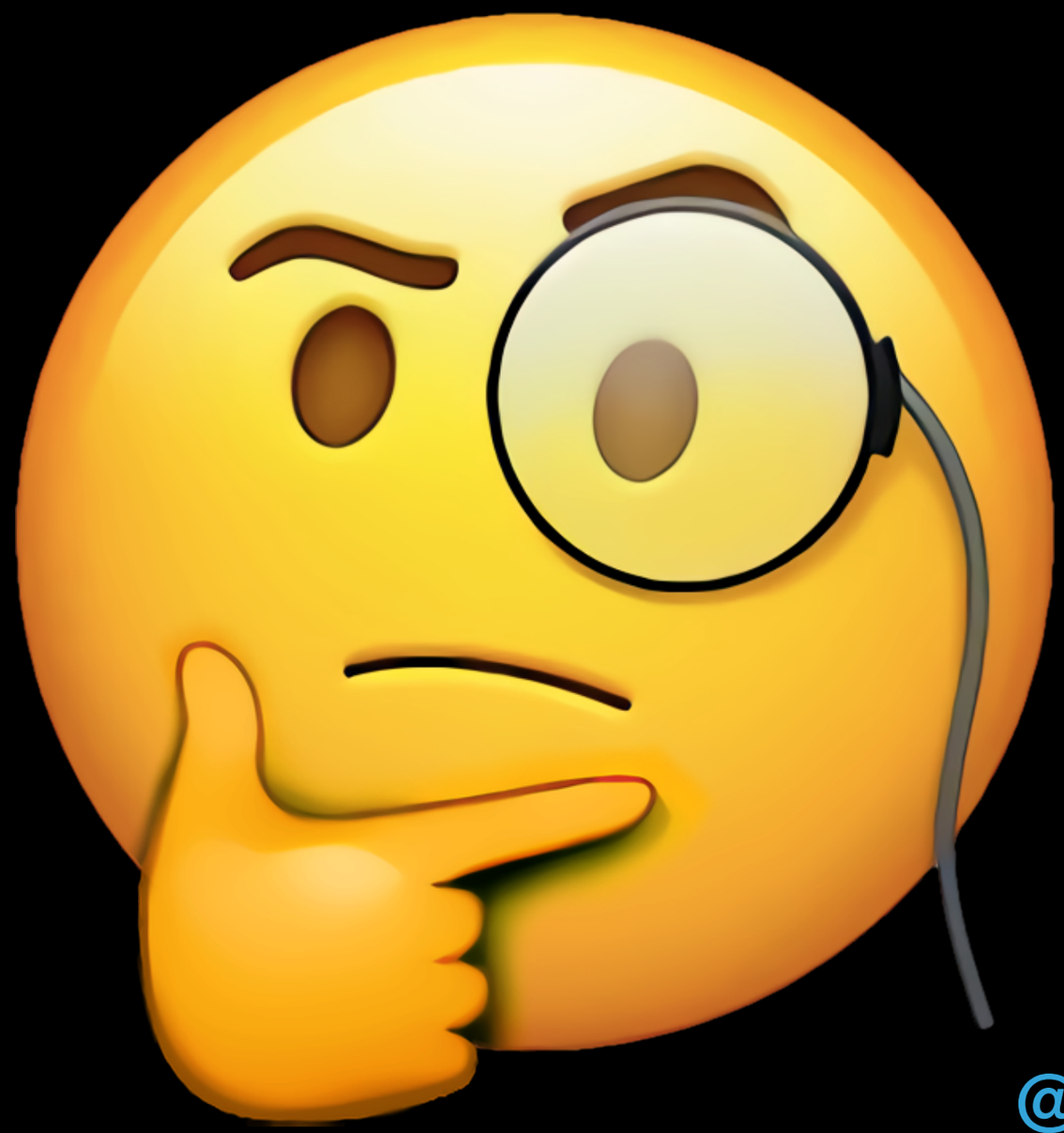
@_jadedickinson

All about flamegraphs

- Why do we need flamegraphs?
- How they were invented
- Alternatives for visualising profiles
- Profiling your own apps

@_jadedickinson

Why do we need flamegraphs?



@_jadedickinson

Why do we need flamegraphs?

Frame Info

/Users/Jade.Dickinson/projects/chopin/app/models/vertical.rb:15:in `Vertical#questionnaire'	(1048 samples - 34.30%)
/Users/Jade.Dickinson/projects/chopin/app/models/rfq.rb:1473:in `Rfq#questionnaire'	(1124 samples - 36.79%)
/Users/Jade.Dickinson/projects/chopin/app/helpers/backoffice/rfqs_helper.rb:28:in `Backoffice::RfqsHelper#select_sof_questions'	(1127 samples - 36.89%)
/Users/Jade.Dickinson/projects/chopin/app/helpers/backoffice/rfqs_helper.rb:20:in `Backoffice::RfqsHelper#sof_answers_in_questionnaire_order'	(504 samples - 16.50%)
/Users/Jade.Dickinson/projects/chopin/app/helpers/backoffice/rfqs_helper.rb:20:in `Backoffice::RfqsHelper#sof_answers_in_questionnaire_order'	(504 samples - 16.50%)
/Users/Jade.Dickinson/projects/chopin/app/presenters/backoffice/base_rfq_presenter.rb:226:in `Backoffice::BaseRfqPresenter#all_sof_answers'	(504 samples - 16.50%)
/Users/Jade.Dickinson/projects/chopin/app/presenters/backoffice/base_rfq_presenter.rb:78:in `Backoffice::BaseRfqPresenter#rfq_details'	(1130 samples - 36.99%)
/Users/Jade.Dickinson/projects/chopin/app/views/backoffice/rfqs/_rfq_details_table.html.haml::in `ActionView::CompiledTemplates#_app_views_backoffice_rfqs__rfq_details_table_html_haml__1517135409571939764_70110887633100'	(1143 samples - 37.41%)
… actionview-5.2.4.1/lib/action_view/template.rb:156:in `ActionView::Template#render'	(2034 samples - 66.58%)
… activesupport-5.2.4.1/lib/active_support/notifications.rb:166:in `#x00007f87f70a1370>.instrument'	(2080 samples - 68.09%)
… actionview-5.2.4.1/lib/action_view/template.rb:353:in `ActionView::Template#instrument_render_template'	(2034 samples - 66.58%)
… actionview-5.2.4.1/lib/action_view/template.rb:156:in `ActionView::Template#render'	(2034 samples - 66.58%)
… actionview-5.2.4.1/lib/action_view/renderer/partial_renderer.rb:332:in `ActionView::PartialRenderer#render_partial'	(1875 samples - 61.37%)
… actionview-5.2.4.1/lib/action_view/renderer/abstract_renderer.rb:40:in `ActionView::AbstractRenderer#instrument'	(1977 samples - 64.71%)
… activesupport-5.2.4.1/lib/active_support/notifications.rb:166:in `#x00007f87f70a1370>.instrument'	(2080 samples - 68.09%)
… activesupport-5.2.4.1/lib/active_support/notifications/instrumenter.rb:19:in `ActiveSupport::Notifications::Instrumenter#instrument'	(2080 samples - 68.09%)
… activesupport-5.2.4.1/lib/active_support/notifications.rb:166:in `#x00007f87f70a1370>.instrument'	(2080 samples - 68.09%)
… actionview-5.2.4.1/lib/action_view/renderer/abstract_renderer.rb:40:in `ActionView::AbstractRenderer#instrument'	(1977 samples - 64.71%)
… actionview-5.2.4.1/lib/action_view/renderer/partial_renderer.rb:332:in `ActionView::PartialRenderer#render_partial'	(1875 samples - 61.37%)
… actionview-5.2.4.1/lib/action_view/renderer/partial_renderer.rb:297:in `ActionView::PartialRenderer#render'	(1901 samples - 62.23%)

@_jadedickinson

How they were invented

- Queue vol. 14 no.2, 2016 - https://queue.acm.org/detail.cfm?id=2927301

visualization 1 OF 28 TEXT ONLY

The Flame Graph

THIS VISUALIZATION OF SOFTWARE EXECUTION IS A NEW NECESSITY FOR PERFORMANCE PROFILING AND DEBUGGING

BRENDAN GREGG, NETFLIX

An everyday problem in our industry is understanding how software is consuming resources, particularly CPUs. What exactly is consuming how much, and how did this change since the last software version? These questions can be answered using software profilers, tools that help direct developers to optimize their code and operators to tune their environment. The output of profilers can be verbose, however, making it laborious to study and comprehend. The flame graph provides a new visualization for profiler output and can make for much faster comprehension, reducing the time for root cause analysis.

In environments where software changes rapidly, such as the Netflix cloud microservice architecture, it is especially important to understand profiles quickly. Faster comprehension can also make the study of foreign software more successful, where one's skills, appetite, and time are strictly limited.

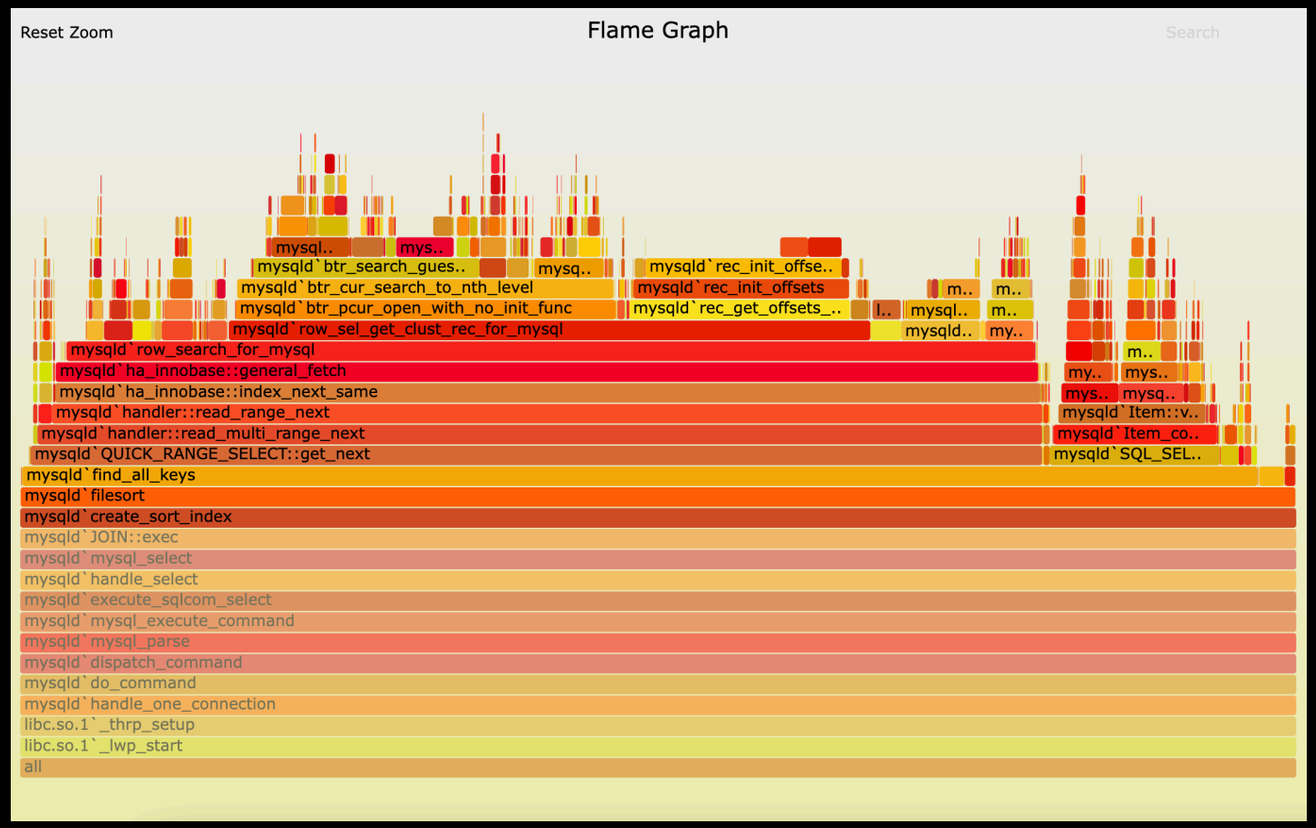
Flame graphs can be generated from the output of many different software profilers, including profiles for different resources and event types. Starting with CPU profiling, this

acmqueue | march-april 2016 1

@_jadedickinson

Alternatives for visualising profiles

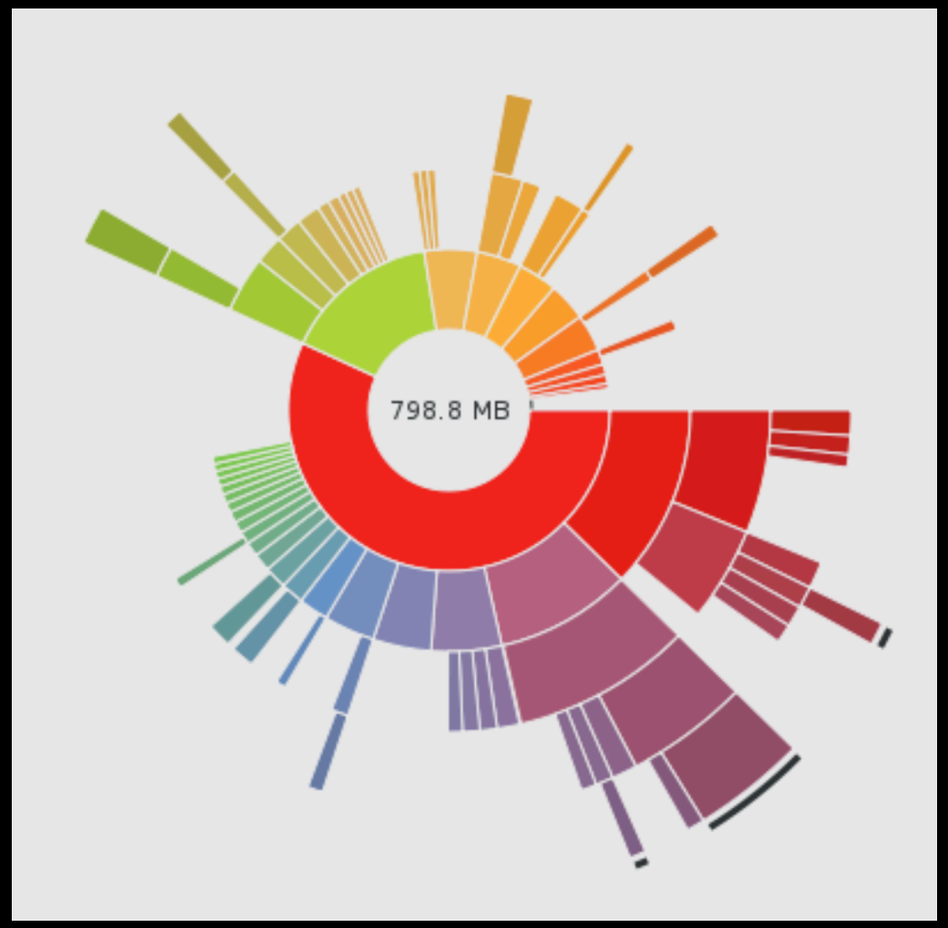
- Profiles sorted alphabetically



@_jadedickinson

Alternatives for visualising profiles

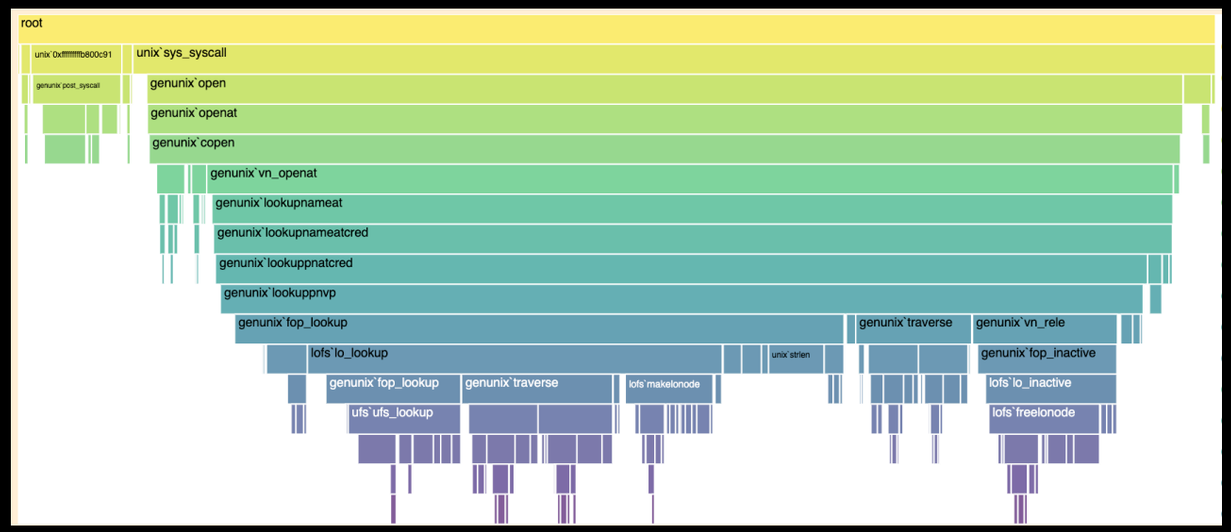
- Profiles sorted alphabetically
- Sunburst charts



@_jadedickinson

Alternatives for visualising profiles

- Profiles sorted alphabetically
- Sunburst charts
- Icicle charts



@_jadedickinson

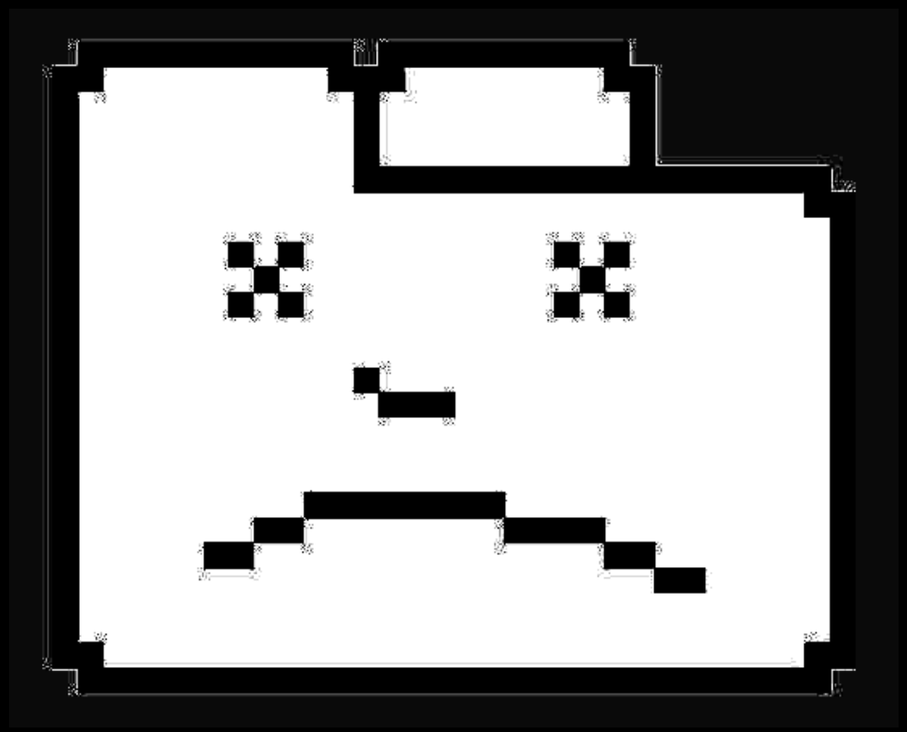
Back to flamegraphs

- Speedscope

@_jadedickinson

Back to flamegraphs

- Speedscope
- Why did rack-mini-profiler need a new renderer



@_jadedickinson

EXERCISE 6

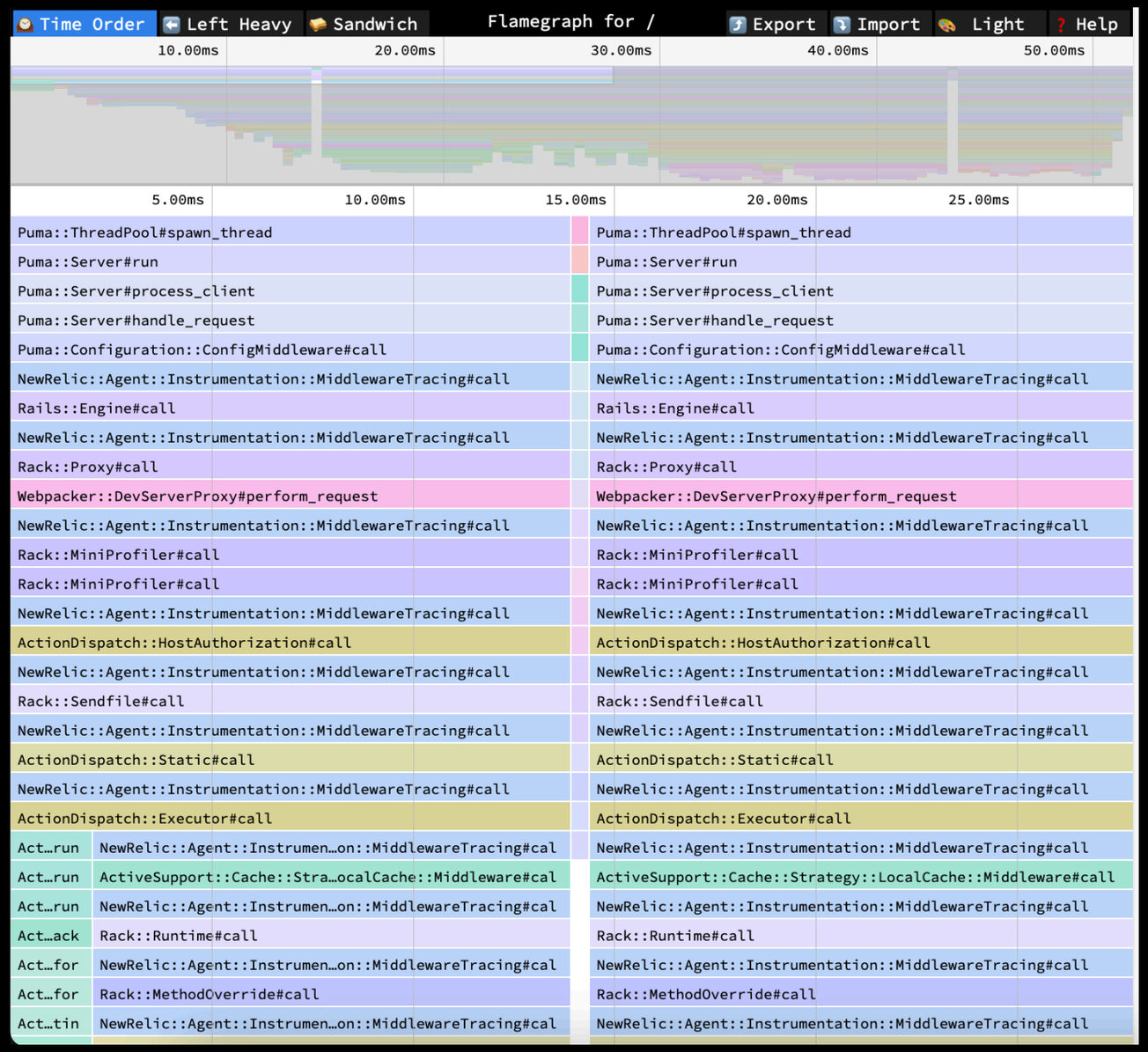
$ Control C
$ git checkout speedscope-renderer
$ bundle install
$ bundle exec rails server

Go to http://localhost:3000/?pp=flamegraph

@_jadedickinson

Speedscope

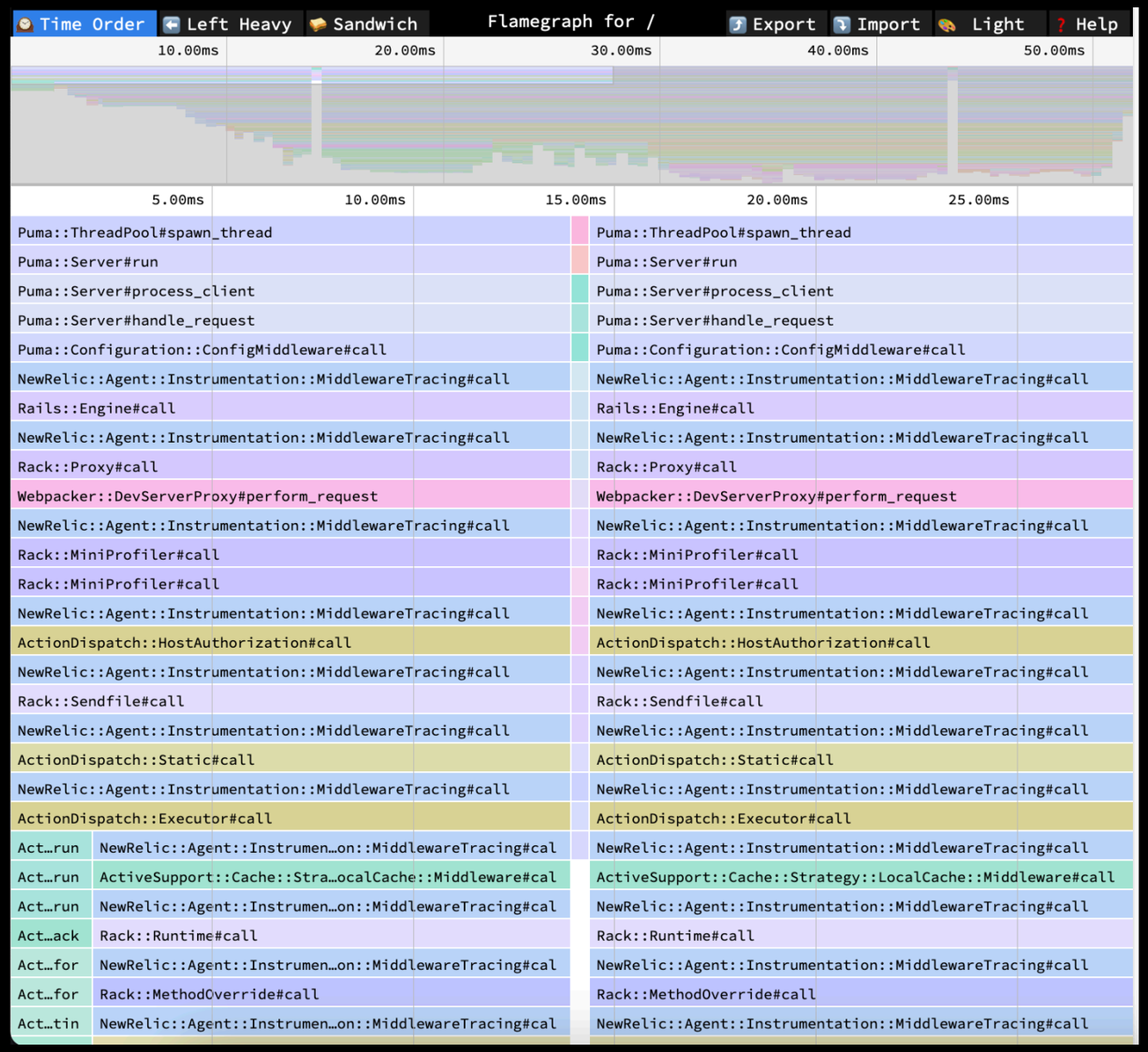
- Performant for larger profiles



@_jadedickinson

Speedscope

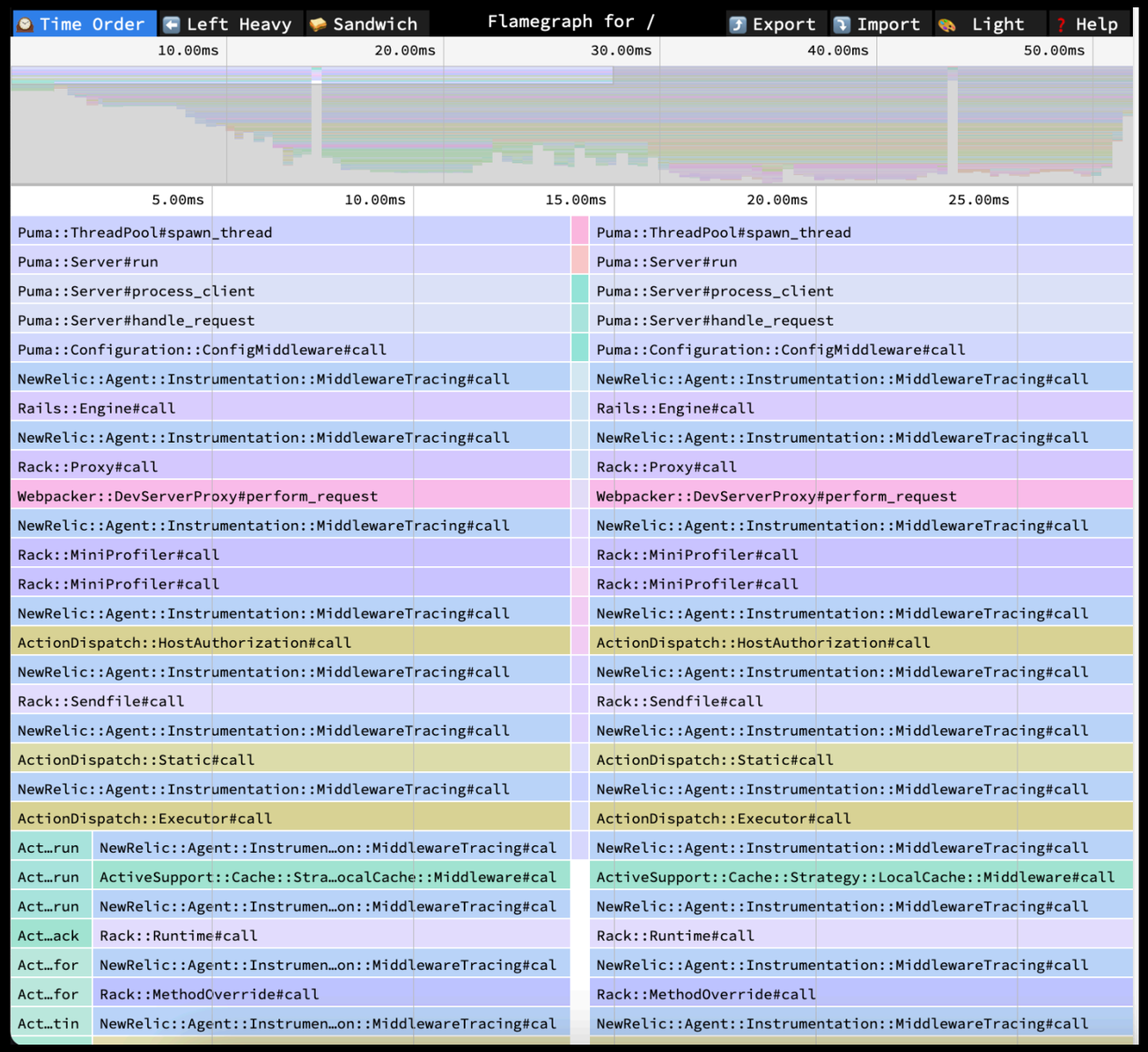
- Performant for larger profiles
- Icicle layout



@_jadedickinson

Speedscope

- Performant for larger profiles
- Icicle layout
- Other options



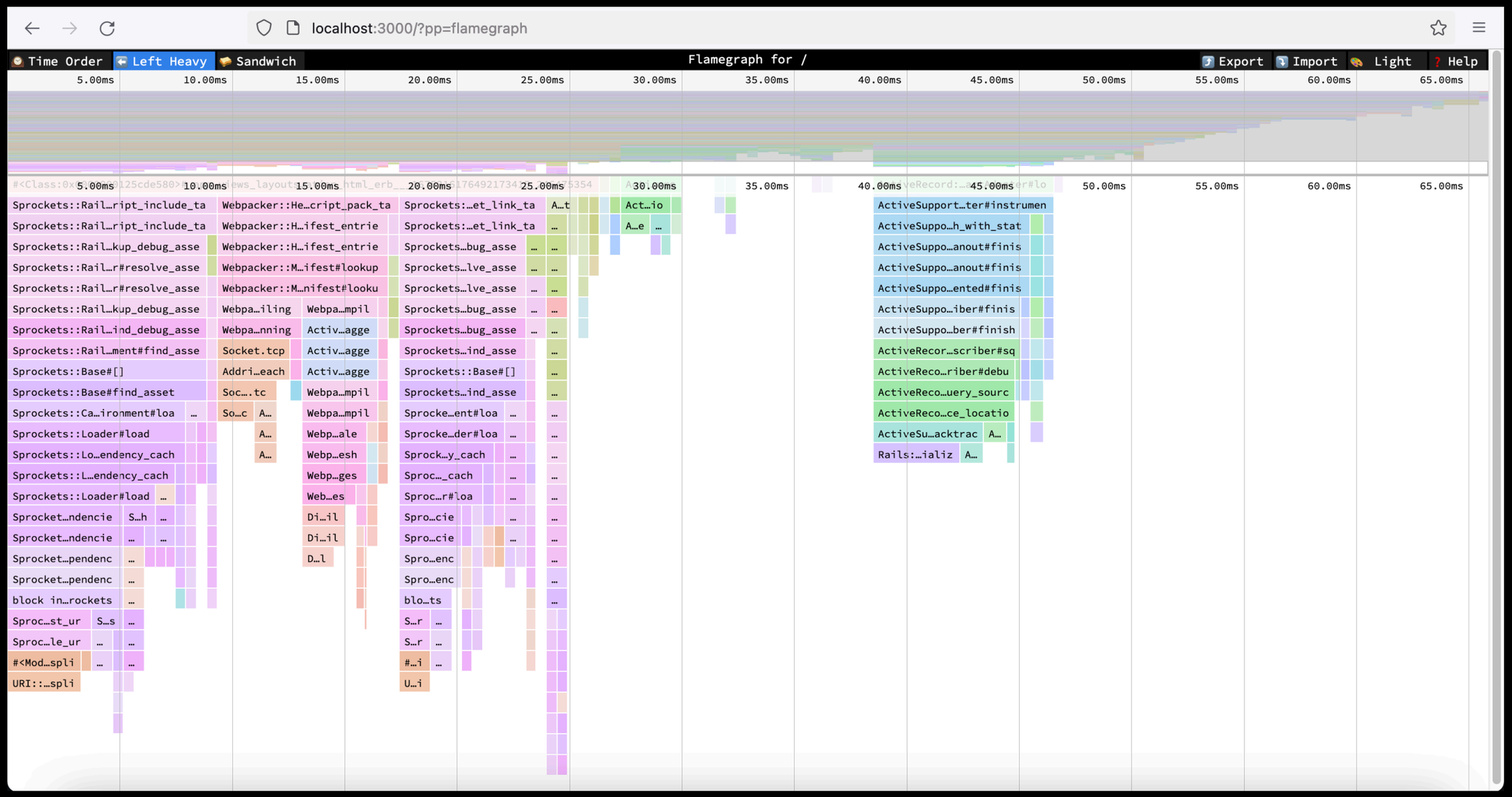
@_jadedickinson

Additional options in Speedscope

- Left Heavy
- Sandwich
- Export and import

@_jadedickinson

Left heavy



@_jadedickinson

Sandwich

localhost:3000/?pp=flamegraph

Time Order | Left Heavy | Sandwich | Flamegraph for / | Export | Import | Light | Help

Total	Self	Symbol Name
5.14ms (7.8%)	5.14ms (7.8%)	URI::RFC3986_Parser#split
4.82ms (7.3%)	4.82ms (7.3%)	Rails::BacktraceCleaner#initialize
2.84ms (4.3%)	2.84ms (4.3%)	ActiveSupport::FileUpdateChecker#watched
2.39ms (3.6%)	2.39ms (3.6%)	Sprockets::PathUtils#stat
2.36ms (3.6%)	2.36ms (3.6%)	ActiveRecord::LogSubscriber#backtrace_cleaner
2.01ms (3.0%)	2.01ms (3.0%)	Logger::LogDevice#write
1.91ms (2.9%)	1.91ms (2.9%)	Digest::Instance#file
1.86ms (2.8%)	1.86ms (2.8%)	#<Module:0x00000001342db948>.generate_guid
2.32ms (3.5%)	1.84ms (2.8%)	Sprockets::URITar#initialize
3.24ms (4.9%)	1.63ms (2.5%)	Socket.tcp
1.57ms (2.4%)	1.57ms (2.4%)	ActionDispatch::FileHandler#file_readable?
1.51ms (2.3%)	1.51ms (2.3%)	ActiveSupport::BacktraceCleaner#add_gem_filter
1.04ms (1.6%)	1.04ms (1.6%)	Addrinfo#connect_internal
963.00µs (1.5%)	963.00µs (1.5%)	Pathname#chop_basename
943.00µs (1.4%)	943.00µs (1.4%)	Sprockets::URITar#expand
938.00µs (1.4%)	938.00µs (1.4%)	ActiveRecord::Base.logger
1.41ms (2.1%)	936.00µs (1.4%)	Sprockets::PathUtils#find_matching_path_for_extensions
902.00µs (1.4%)	902.00µs (1.4%)	block in attribute_aliases=
607.00µs (0.92%)	607.00µs (0.92%)	PG::Connection#exec_prepared
3.24ms (4.9%)	574.00µs (0.87%)	Addrinfo.foreach
529.00µs (0.80%)	529.00µs (0.80%)	ActiveSupport::HashWithIndifferentAccess#[]
520.00µs (0.79%)	520.00µs (0.79%)	PG::Connection#exec_params
513.00µs (0.78%)	513.00µs (0.78%)	#<Module:0x0000000153c8c130>.touch
512.00µs (0.78%)	512.00µs (0.78%)	ActiveSupport::Callbacks::CallbackSequence#final?
506.00µs (0.77%)	506.00µs (0.77%)	#<Module:0x0000000136232378>.all_view_paths
501.00µs (0.76%)	501.00µs (0.76%)	#<Module:0x0000000153939d48>.encode_www_form_component
500.00µs (0.76%)	500.00µs (0.76%)	ActionDispatch::Request::Utils.check_param_encoding
498.00µs (0.76%)	498.00µs (0.76%)	IPAddr#initialize
492.00µs (0.75%)	492.00µs (0.75%)	ActiveRecord::LogSubscriber#sql_color
490.00µs (0.74%)	490.00µs (0.74%)	#<Module:0x0000000156ba9e90>.map_product
65.46ms (>99%)	490.00µs (0.74%)	NewRelic::Agent::Instrumentation::MiddlewareTracing#call
1.37ms (2.1%)	489.00µs (0.74%)	ActiveRecord::PredicateBuilder#expand_from_hash
487.00µs (0.74%)	487.00µs (0.74%)	Concurrent::Collection::NonConcurrentMapBackend#[]
487.00µs (0.74%)	487.00µs (0.74%)	ActiveSupport::ParameterFilter::CompiledFilter#value_for_key

@_jadedickinson

Export and import

JSON **JSONLint** - The JSON Validator Try the New Pro More Developer Tools


```
1  {
2      "exporter": "speedscope@1.12.1",
3      "name": "Flamegraph for /",
4      "activeProfileIndex": 0,
5      "$schema": "https://www.speedscope.app/file-format-schema.json",
6      "shared": {
7          "frames": [{
8              "name": "Puma::ThreadPool#spawn_thread",
9              "file": "/opt/homebrew/Cellar/rbenv/1.1.2/versions/2.6.6/lib/ruby/gems/2.6.0/gems/puma-4.3.7/lib/puma/
10             "line": 86
11         }, {
12             "name": "Puma::Server#run",
13             "file": "/opt/homebrew/Cellar/rbenv/1.1.2/versions/2.6.6/lib/ruby/gems/2.6.0/gems/puma-4.3.7/lib/puma/
14             "line": 282
15         }, {
16             "name": "Puma::Server#process_client",
17             "file": "/opt/homebrew/Cellar/rbenv/1.1.2/versions/2.6.6/lib/ruby/gems/2.6.0/gems/puma-4.3.7/lib/puma/
18
```


@_jadedickinson

EXERCISE 7

Play around with Speedscope and take a look at the options.

Try the Left Heavy and Sandwich views

Try exporting a profile to JSON.

@_jadedickinson

Recap

- Reasons we care about performance
- How to prioritise and rule out other causes
- Reading flamegraphs
- Changing some code to speed it up
- Getting that change into production
- Adapting to what stage your work is in
- Speedscope visualiser and where it's most useful

@_jadedickinson

FINAL EXERCISE

Install rack-mini-profiler in your application

https://github.com/JadeDickinson/rubyconf_2021_flamegraphs#installation-of-rack-mini-profiler-with-default-speedscope-renderer-for-your-own-project

Go through the steps we've talked about

Can you find something to speed up?

@_jadedickinson

REFERENCES

- Nate Berkopec: https://www.speedshop.co/blog/
- Julia Evans:
- https://jvns.ca/juliasections/rbspy/
- https://jvns.ca/blog/2016/02/10/have-high-expectations-for-computers/
- https://jvns.ca/blog/2017/12/19/how-much-does-the-ruby-abi-change-/
- https://jvns.ca/blog/2017/12/17/how-do-ruby---python-profilers-work-/
- Pat Shaughnessy: http://patshaughnessy.net/ & Ruby Under a Microscope
- https://jemma.dev/blog/gc-incremental
- https://jemma.dev/blog/gc-generational

@_jadedickinson

REFERENCES CONTINUED

- Profilers:
- https://github.com/MiniProfiler/rack-mini-profiler
- Stackprof: https://github.com/tmm1/stackprof
- Rubyprof: https://ruby-prof.github.io/
- rbspy: https://rbspy.github.io/
- https://azukidigital.com/blog/2015/json-rack-mini-profiler/
- Java: https://www.brendangregg.com/flamegraphs.html & https://www.youtube.com/watch?v=D53T1Ejig1Q
- Python: https://eng.uber.com/pyflame-python-profiler/
- Node.js: https://nodejs.org/en/docs/guides/diagnostics-flamegraph/
- Elm: https://blog.swmansion.com/hunting-js-memory-leaks-in-react-native-apps-bd73807d0fde
- React: https://shakacode.gitbooks.io/react-on-rails/content/ and https://youtu.be/xsSnOQynTHs
- React Native: https://blog.swmansion.com/hunting-js-memory-leaks-in-react-native-apps-bd73807d0fde
- https://www.rawsignal.ca/books

@_jadedickinson

EVEN MORE REFERENCES

- https://www.brendangregg.com/flamegraphs.html
- https://queue.acm.org/detail.cfm?id=2927301
- https://github.com/brendangregg/FlameGraph
- https://www.usenix.org/conference/lisa10/visualizations-performance-analysis-and-more
- https://www.usenix.org/legacy/events/lisa10/tech/slides/gregg.pdf
- https://github.com/SamSaffron/flamegraph
- https://samsaffron.com/archive/2013/03/19/flame-graphs-in-ruby-miniprofiler
- Speedscope:
- http://jamie-wong.com/post/speedscope/
- https://www.speedscope.app/
- https://github.com/jlfwong/speedscope#usage
- https://johnysswlab.com/speedscope-visualize-what-your-program-is-doing-and-where-it-is-spending-time/
- https://hacks.mozilla.org/2018/11/cross-language-performance-profile-exploration-with-speedscope/

@_jadedickinson

THANK YOU

- @_jadedickinson on Twitter, jadedickinson.com
- Want to hear from you after - what did you find, did you speed anything up?

@_jadedickinson