PROFILING YOUR STACK

EVERYTHING'S SLOW, WHAT NOW?

Trey Pendragon
Princeton University Library

STEP 1: INITIAL TEST

Profiling and fixing performance bugs is time-intensive – a quick test in Chrome should tell you if it's worth exploring.

Open the network tab of the developer tools, scroll to the top, and do your action.

JUNE 21 - UGH

tpendragon 12:15 PM
File manager takes a long time to load..

Trying to persist a reorder.

It's uh

Taking a while

tpendragon 12:25 PM
Oh it finished

8.3 minutes

escowles 12:26 PM
so that's less than ideal...

tpendragon 12:26 PM ☆
Yeah..

STEP 2: FIND WHERE TO FOCUS

A request is slow, but what part of the interaction is the slow part?

- **Disk I/O?**
- **Database writes?**
- **Queries?**
- **Ruby code?**
- **Network?**

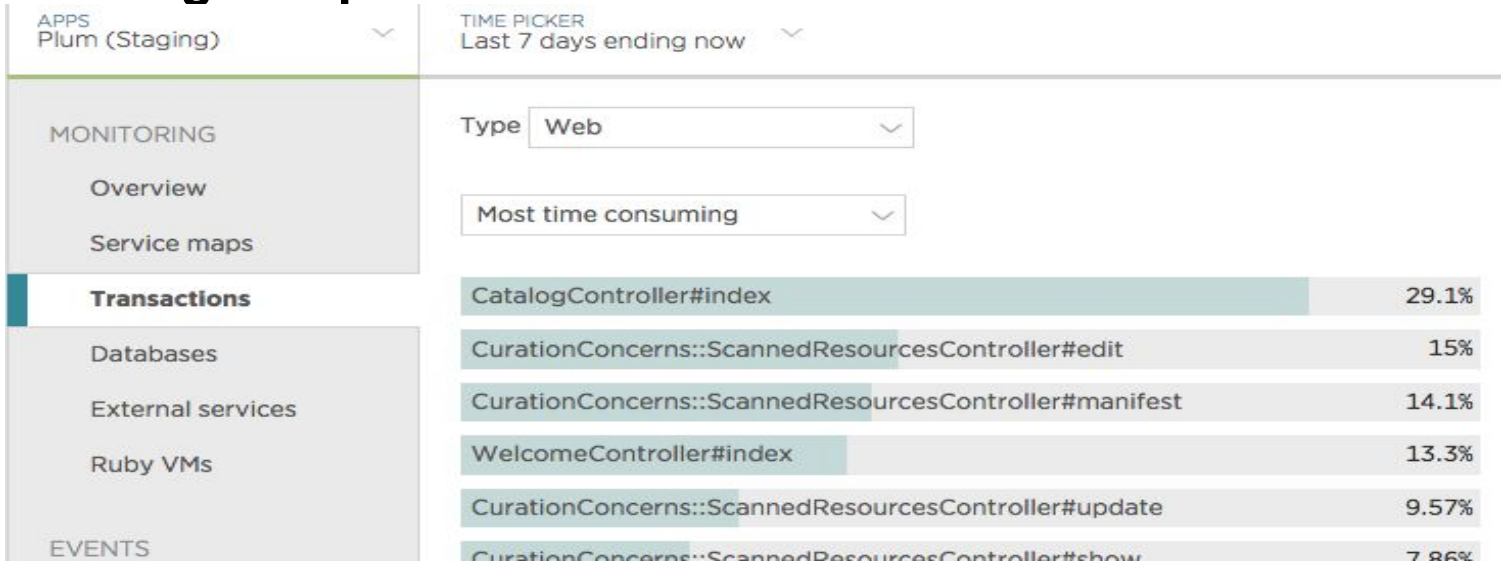
STEP 2: FIND WHERE TO FOCUS

Have response monitoring infrastructure up EARLY. Any tool which tracks the request/response cycle saves a mountain of time.

We use New Relic

NEW RELIC

- **View transactions by most time consuming or slowest average response time.**

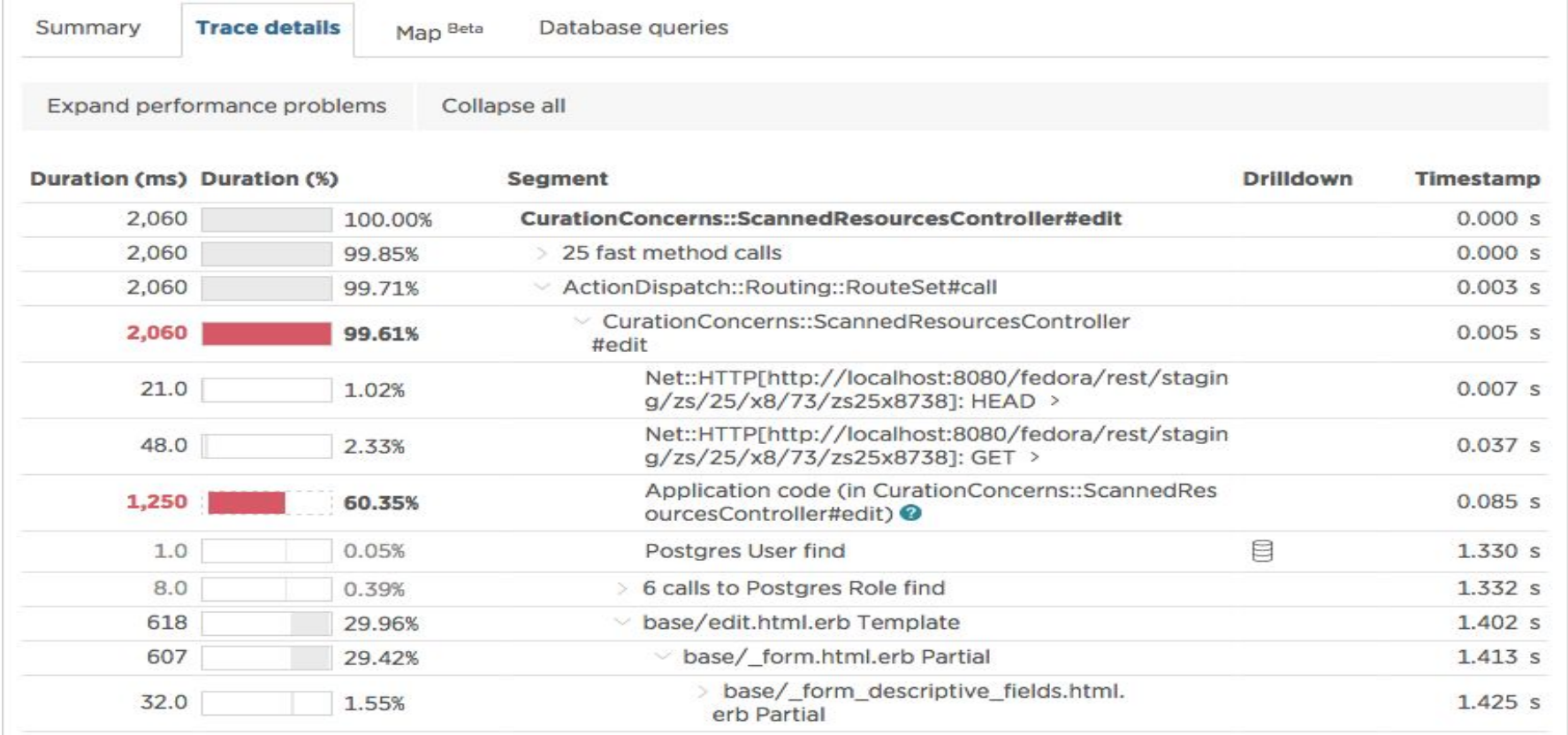


NEW RELIC

- Click the relevant transaction
- Look at a trace

NEW RELIC

- **View trace details**



Duration (ms)	Duration (%)	Segment	Drilldown	Timestamp
2,060	100.00%	**CurationConcerns::ScannedResourcesController#edit**		0.000 s
2,060	99.85%	25 fast method calls		0.000 s
2,060	99.71%	ActionDispatch::Routing::RouteSet#call		0.003 s
2,060	**99.61%**	CurationConcerns::ScannedResourcesController #edit		0.005 s
21.0	1.02%	Net::HTTP[http://localhost:8080/fedora/rest/staging/zs/25/x8/73/zs25x8738]: HEAD		0.007 s
48.0	2.33%	Net::HTTP[http://localhost:8080/fedora/rest/staging/zs/25/x8/73/zs25x8738]: GET		0.037 s
1,250	**60.35%**	Application code (in CurationConcerns::ScannedResourcesController#edit)		0.085 s
1.0	0.05%	Postgres User find		1.330 s
8.0	0.39%	6 calls to Postgres Role find		1.332 s
618	29.96%	base/edit.html.erb Template		1.402 s
607	29.42%	base/_form.html.erb Partial		1.413 s
32.0	1.55%	base/_form_descriptive_fields.html.erb Partial		1.425 s

STEP 3: PROFILING RUBY (IF NECESSARY)

- You're probably past the point of the Benchmark module providing enough information now.
- Enter RubyProf (https://github.com/ruby-prof/ruby-prof)

```
15 require 'ruby-prof'
14 desc "Profiles saving a resource"
13 task profile_saving: :environment do
12   s = ScannedResource.find("z603qz15t")
11   s.title = ["Profile Testing"]
10   puts "Running Profile"
 9   result = RubyProf.profile do
 8     s.save!
 7   end
 6   printer = RubyProf::CallStackPrinter.new(result)
 5   printer.print(File.open("tmp/test_dump.html", 'w'), min_percent: 1)
 4 end
 3
 2
```

STEP 3: PROFILING RUBY

23,543 calls to RDFSource#get_values was our culprit.

That many calls to ANY code is going to be slow.

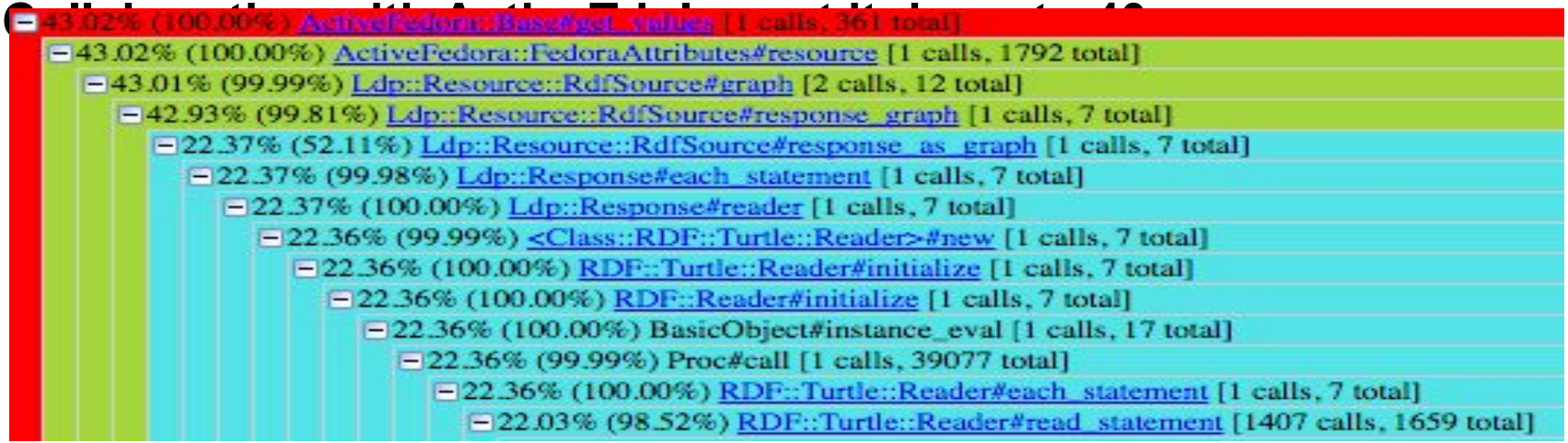
STEP 4: CALL IN REINFORCEMENTS

You have enough research now to get a lot of help if you need it. Message the community – you'll get a response.

There would be a picture of a bunch of collaboration on Slack here if the logs went back that far.

STEP 5: FIX IT AND START OVER

It turned out ActiveFedora was copying graph objects statement-by-statement many times it didn't need to. One PR cut things down to around 60 seconds to persist.



WHAT IF IT'S FEDORA?

After we fixed the ruby problems, New Relic showed that GETs of a single 1,000 page object were taking 40-60 seconds.

REPRODUCTION

Get the smallest set of test scripts you can to reproduce the problem (thanks Esmé!) and either message the Fedora community on IRC or make a ticket on their Jira.

- **https://jira.duraspace.org/projects/FCREPO/issues/**

GET INVOLVED

Fedora's always looking for community involvement. If you feel like you can contribute towards your own performance issues, please get involved!

MORAL OF THE STORY

Profiling is often difficult and exhausting. It takes a lot of time.

- **Decide if it's worth the investment.**
- **Focus your effort.**
- **Spend time on the slowest pieces first. Sometimes it's easy to work on micro-improvements – avoid the temptation.**

QUESTIONS?

Trey Pendragon

tpendragon@princeton.edu

@pendragon_dt