

Speeding up RSpec tests in a large Rails application

Asked 14 years, 3 months ago Modified 4 years, 7 months ago

Viewed 20k times



96



I have a Rails application with over 2,000 examples in my RSpec tests. Needless to say, it's a large application and there's a lot to be tested. Running these tests at this point is very inefficient and because it takes so long, we're almost at the point of being discouraged from writing them before pushing a new build. I added `--profile` to my `spec.opts` to find the longest running examples and there are at least 10 of them that take an average of 10 seconds to run. Is that normal amongst you RSpec experts? Is 10 seconds entirely too long for one example? I realize that with 2,000 examples, it will take a non-trivial amount of time to test everything thoroughly - but at this point 4 hours is a bit ludicrous.

What kind of times are you seeing for your longest running examples? What can I do to troubleshoot my existing specs in order to figure out bottlenecks and help speed things up. Every minute would really help at this point.

ruby-on-rails

ruby

unit-testing

rspec

bdd

Share

Improve this question

Follow

asked Sep 7, 2010 at 21:58



randombits

48.4k ● 79 ● 271 ● 447

1 Are the slow tests integration tests? Are they hitting a db? If so, how often is the db being reloaded and can you mock the db? – [Kaleb Pederson](#) Sep 7, 2010 at 22:44

1 Are you able to just run part of the specs that are relevant to what part you're working on, akin to SeattleRB's autotest? Do you have a continuous integration server that can run all of the tests? – [Andrew Grimm](#) Sep 7, 2010 at 22:55

Remeber too that all things are relative. I've heard "grrr, our test suite takes forever" for both 20 minutes... and 16-20 hours. It's all in the eye of the beholder. 10 seconds for a given tests often means a unit test that has become an integration test as mentioned below. – [Michael Durrant](#) Apr 14, 2013 at 16:43

A suggestion for this kind of problem: use `perftools.rb` together with your test framework to understand what's using up most of your time. Take the top 10 calls and try to eliminate/skim them. Then repeat, until happy.
– [aledalgrande](#) Feb 25, 2014 at 20:44

7 Answers

Sorted by:

Highest score (default)



123



10 seconds is a very long time for any single test to run. My gut feeling is that your spec target is running both unit and integration tests at the same time. This is a typical thing that projects fall into and at some stage, you *will need to overcome this technical debt* if you want to produce more, faster. There are a number of strategies



which can help you to do this... and I'll recommend a few that I have used in the past.



1. Separate Unit From Integration Tests

The first thing I would do is to separate unit from integration tests. You can do this either by:

1. Moving them (into separate folders under the spec directory) - and modifying the rake targets
2. Tagging them (rspec allows you to tag your tests)

The philosophy goes, that you want your regular builds to be quick - otherwise people won't be too happy to run them often. So get back to that territory. Get your regular tests to run quick, and use a continuous integration server to run the more complete build.

An integration test is a test that involves external dependencies (e.g. Database, WebService, Queue, and some would argue FileSystem). A unit test just tests the specific item of code that you want checked. It should run fast (9000 in 45 secs is possible), i.e. most of it should run in memory.

2. Convert Integration Tests To Unit Tests

If the bulk of your unit tests is smaller than your integration test suite, you have a problem. What this means is that inconsistencies will begin to appear more easily. So from here, start creating more unit tests to

replace integration tests. Things you can do to help in this process are:

1. Use a mocking framework instead of the real resource. Rspec has an inbuilt mocking framework.
2. Run rcov on your unit test suite. Use that to gauge how thorough your unit test suite is.

Once you have a proper unit test(s) to replace an integration test - remove the integration test. Duplicate testing only makes maintenance worse.

3. Don't Use Fixtures

Fixtures are evil. Use a factory instead (machinist or factorybot). These systems can build more adaptable graphs of data, and more importantly, they can build in-memory objects which you can use, rather than load things from an external data source.

4. Add Checks To Stop Unit Tests Becoming Integration Tests

Now that you have faster testing in place, time to put in checks to STOP this from occurring again.

There are libraries which monkey patch ActiveRecord to throw an error when trying to access the database (UnitRecord).

You could also try pairing and TDD which can help force your team to write faster tests because:

1. Somebody's checking - so nobody gets lazy
2. Proper TDD requires fast feedback. Slow tests just make the whole thing painful.

5. Use Other Libraries To Overcome The Problem

Somebody mentioned `spork` (speeds up load times for the test suite under rails3), `hydra/parallel_tests` - to run unit tests in parallel (across multiple cores).

This should probably be used LAST. Your real problem is all the way in step 1, 2, 3. Solve that and you will be in a better position to role out additional infrastructure.

Share Improve this answer

Follow

edited Apr 27, 2020 at 20:14



sameers

5,085 ● 3 ● 38 ● 45

answered Feb 23, 2011 at 0:43



BlueFish

5,135 ● 3 ● 28 ● 36

Great answer. It is very true that no advanced tools can help run bad tests quickly. So try writing only good ones.

– [Yura Omelchuk](#) Dec 6, 2011 at 22:47

-
- 5 I disagree on your third point that you should not use fixtures. If used properly they are faster than factories as you are not having to create objects. Your data is there you don't have to create it with each test case. – [Jacob Waller](#) Feb 14, 2014 at 14:15

-
- 3 This about factories being faster is a joke, right?
– [Mike Szyndel](#) Mar 26, 2014 at 16:41
-



18

For a great cookbook on improving the performance of your test suite, check out the [Grease Your Suite](#) presentation.



He documents a 45x speedup in test suite run time by utilizing techniques such as:



- [fast_context](#)
- [quickerclip](#)
- [hydra](#)
- scary file system tweaks
- [tcmalloc](#)
- [Ruby Enterprise Edition GC tuning](#)

Share Improve this answer

answered Sep 8, 2010 at 13:29

Follow



[marshally](#)

3,567 ● 2 ● 26 ● 26

I like the links, but not the presentation. Index cards for someone's speech are a prompt for the speaker to fill in the meat of the presentation. – [Brian Maltzan](#) Sep 8, 2010 at 13:51

This presentation is no longer up. fast_context speeds up multiple shoulda blocks. quickerclip (that link also dead) apparently speeds up Paperclip. Hydra is deprecated for parallel_tests and Gorgon. bootsnap has file system tweaks

and is now included with Rails. Recent versions of Ruby have improved allocation and GC. – [rep](#) Jun 22, 2018 at 19:49



5



You can use Spork. It has support for 2.3.x ,

<https://github.com/sporkrb/spork>

or ./script/spec_server which may work for 2.x

Also you can edit the database configuration (which essentially speeds up the database queries etc), which will also increase performance for tests.

Share Improve this answer

Follow

edited Apr 11, 2013 at 17:48



Pigueiras

19.3k ● 10 ● 66 ● 87

answered Sep 7, 2010 at 22:09




Rishav Rastogi

15.5k ● 3 ● 43 ● 47

Spork is awesome. It also works on Rails 3, with a bit of hacking. But one thing to note with Spork on Rails 3 is that it doesn't seem to pick up changes in controllers, so you'll need to restart it every so often. But anyway, Spork is really awesome, the speed improvement of the tests is very significant. – [AboutRuby](#) Sep 8, 2010 at 0:18 ✎

4 Spork is for speeding up startup only, not tests. So with a lot of specs, the advantage is insignificant. – [Reactormonk](#) Sep 8, 2010 at 8:19

Contrary to above comments spork can be used to speed up tests as well as startup as per railscast

railscasts.com/episodes/285-spork. Massively improved my rspec test suite on a very large app. (Reduced from 192 seconds to 10 seconds!) – [jamesc](#) Nov 26, 2011 at 10:55 



4



10 seconds per example seems like a very long time. I've never seen a spec that took more than one second, and most take far less. Are you testing network connections? Database writes? Filesystem writes?

Use mocks and stubs as much as possible - they are *much* faster than writing code that hits the database. Unfortunately mocking and stubbing also take more time to write (and are harder to do correctly). You have to balance the time spent writing tests vs. the time spent running tests.

I second [Andrew Grimm](#)'s comment about looking into a CI system which might allow you to parallelize your test suite. For something that size, it might be the only viable solution.

Share Improve this answer

Follow

edited May 23, 2017 at 11:53



Community Bot

1 • 1


answered Sep 8, 2010 at 8:08



[zetetic](#)

47.5k • 10 • 113 • 119

yeah if it's 10 seconds I would profile it and see where the bottleneck is – [rogerdpack](#) Sep 8, 2010 at 22:49

1 I have a huge project and running a single rspec test takes near 15-20 seconds. – [ExiRe](#) Apr 11, 2013 at 11:30 



If you are using ActiveRecord models, you should also consider the cost of BCrypt encryption.

2

You can read more about it on this blog post:



<http://blog.syncopelabs.co.uk/2012/12/speed-up-rspec-test.html>



Share Improve this answer

answered Dec 21, 2012 at 22:43

Follow



[Erdem Gezer](#)

3,272 ● 2 ● 23 ● 16



faster_require gem might help you. Besides that your only way is to (like you did) profile and optimize, or use spork or something that runs your specs in parallel for you.

1



http://ruby-toolbox.com/categories/distributed_testing.html



Share Improve this answer

edited Sep 8, 2010 at 22:50

Follow

answered Sep 8, 2010 at 3:04



[rogerdpack](#)

66.5k ● 39 ● 282 ● 401



Delete the existing test suite. Will be incredibly effective.

-6

Share Improve this answer

answered Feb 5, 2016 at 0:44

Follow



thedanotto

7,287 ● 5 ● 48 ● 46

