Unit test execution speed (how many tests per second?)

Asked 16 years, 4 months ago   Modified 4 years, 9 months ago

Viewed 20k times

27

What kind of execution rate do you aim for with your unit tests (# test per second)? How long is too long for an individual unit test?

I'd be interested in knowing if people have any specific thresholds for determining whether their tests are too slow, or is it just when the friction of a long running test suite gets the better of you?

Finally, when you do decide the tests need to run faster, what techniques do you use to speed up your tests?

Note: integration tests are obviously a different matter again. We are strictly talking unit tests that need to be run as frequently as possible.

Response roundup: Thanks for the great responses so far. Most advice seems to be don't worry about the speed -- concentrate on quality and just selectively run them if they are too slow. Answers with specific numbers have included aiming for <10ms up to 0.5 and 1 second per test, or just keeping the entire suite of commonly run tests under 10 seconds.

Not sure whether it's right to mark one as an "accepted answer" when they're all helpful :)

unit-testing performance

Share

Improve this question

Follow

edited Jan 9, 2009 at 1:32

asked Aug 13, 2008 at 23:36

David Tchepak

10.4k ● 2 ● 62 ● 72

1 second per test means your test suite will quickly reach the point where you stop running it all the time because it feels too slow. When you you can run 100 tests/sec you'll run the suite much more frequently than when it takes 100 times as long. – Jeffrey Fredrick Jan 9, 2009 at 5:30

10 Answers

Sorted by: Highest score (default)

30

All unit tests should run in under a second (that is all unit tests combined should run in 1 second). Now I'm sure this has practical limits, but I've had a project with a 1000 tests that run this fast on a laptop. You'll really want this speed so your developers don't dread refactoring some core part of the model (i.e., Lemme go get some coffee while I run these tests...10 minutes later he comes back).

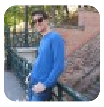
This requirement also forces you to design your application correctly. It means that your domain model is pure and contains zero references to any type of persistance (File I/O, Database, etc). Unit tests are all about testing those business relatonships.

Now that doesn't mean you ignore testing your database or persistence. But these issues are now isolated behind repositories that can be separately tested with integration tests that is located in a separate project. You run your unit tests constantly when writing domain code and then run your integration tests once on check in.

Share Improve this answer Follow

edited May 15, 2018 at 1:45
Community Bot
1 ● 1

answered Aug 14, 2008 at 0:08



Jim
3,170 ● 4 ● 32 ● 38

21 What language, compiler and test engine are you using? 1000 tests per second is insane - I have a hard time getting any better than 4 Tests/Second in C#/Visual Studio 2010.
– Nilzor Mar 18, 2011 at 8:23

I suspect (haven't looked for proof) that the popular test runners in VS don't have much effort put into optimising their speed. 4 tests a second is very slow. If you're going to have a large suite of tests, you want to be able to churn through hundreds per second. Having them all run in a second is not a scalable requirement for large development projects.
– Niall Connaughton Jun 11, 2013 at 1:17

We just made a minor change which made tests run for 6 minutes instead of 2 hours. I see now that if tests were faster, people would be more inclined to work on the component! – Tvde1 Jul 19, 2019 at 10:08

5

The goal is 100s of tests per second. The way you get there is by following Michael Feather's rules of unit tests.

An important point that came up in a past CITCON discussion is that if your tests aren't this fast it is quite likely that you aren't getting the design benefits of unit testing.

Share Improve this answer Follow

answered Nov 20, 2008 at 17:05

Jeffrey Fredrick
4,503 ● 1 ● 27 ● 21

4

If we're talking strictly unit tests, I'd aim more for completeness than speed. If the run time starts to cause friction, separate the test into different project/classes etc., and only run the tests related to what you're working on. Let the Integration server run all the tests on checkin.

Share Improve this answer Follow

answered Aug 13, 2008 at 23:44

Lance Fisher
25.8k ● 23 ● 98 ● 122

I tend to focus more on readability of my tests than speed. However, I still try to make them reasonably fast. I

1

think if they run on the order of milliseconds, you are fine. If they run a second or more per test... then you might be doing something that should be optimized.

Slow tests only become a problem as the system matures and causes the build to take hours, at which point you are more likely running into an issue of a lot of kind of slow tests rather than one or 2 tests that you can optimize easily... thus you should probably pay attention RIGHT AWAY if you see lots of tests running hundreds of milliseconds each (or worse, seconds each), rather than wait till it gets to the hundreds of tests taking that long point (at which point it is going to be really hard to solve the problem).

Even so, it will only reduce the time between when your automated build issues errors... which is ok if it is an hour later (or even a few hours later), I think. The problem is running them before you check in, but this can be avoided by selecting a small subset of tests to run that are related to what you are working on. Just make sure to fix the build if you check in code that breaks tests you didn't run!

Share Improve this answer Follow

answered Aug 13, 2008 at 23:44

Mike Stone
44.6k ● 30 ● 114 ● 140

We're currently at 270 tests in around 3.something seconds. There are probably around 8 tests that perform file IO.

1

These are run automatically upon a successful build of our libraries on every engineers machine. We have more extensive (and time consuming) smoke-testing that is done by the build machine every night, or can be started manually on an engineers machine.

As you can see we haven't yet reached the problem of tests being too time consuming. 10 seconds for me is the point where it starts to become intrusive, when we start to approach that it'll be something we'll take a look at. We'll likely move the lower level libraries, which are more robust since they change infrequently and have few dependencies, into the nightly builds, or a configuration where they're only executed by the build machine.

If you find it's taking more than a few seconds to run a hundred or so tests you may need to examine what you are classifying as a unit test and whether it would be better treated as a smoke test.

your mileage will obviously be highly variable depending on your area of development.

Share Improve this answer Follow

answered Aug 13, 2008 at 23:48

Andrew Grant
58.8k ● 22 ● 131 ● 144

1

Data Point -- Python Regression Tests

Here are the numbers on my laptop for running "make test" for Python 2.5.2:

- number of tests: 3851 (approx)
- execution time: 9 min, 6 sec
- execution rate: 7 tests / sec

Share Improve this answer Follow

answered Aug 14, 2008 at 5:59

Mark Harrison
304k ● 131 ● 350 ● 489

1

One of the most important rules about unit tests is they should run ***fast***.

> How long is too long for an individual unit test?

Developers should be able to run the whole suite of unit tests in seconds, and definitely not in minutes and minutes. Developers should be able to quickly run them after changing the code in anyway. If it takes too long, they won't bother running them and you lose one of the main benefits of the tests.

> What kind of execution rate do you aim for with your unit tests (# test per second)?

You should aim for each test to run in an order of milliseconds, anything over 1 second is probably testing too much.

We currently have about 800 tests that run in under 30 seconds, about 27 tests per second. This includes the time to launch the mobile emulator needed to run them. Most of them each take 0-5ms (if I remember correctly).

We have one or two that take about 3 seconds, which are probably candidates for checking, but the important thing is the whole test suite doesn't take so long that it puts off developers running it, and doesn't significantly slow down our continuous integration build.

We also have a configurable timeout limit set to 5 seconds -- anything taking longer will fail.

Share Improve this answer Follow

answered Sep 17, 2011 at 8:33

Hugo
29.3k ●9 ●85 ●101

0

I judge my unit tests on a per test basis, not by by # of tests per second. The rate I aim for is 500ms or less. If it is above that, I will look into the test to find out why it is taking so long.

When I think a test is to slow, it usually means that it is doing too much. Therefore, just refactoring the test by splitting it up into more tests usually does the trick. The other times that I have noticed my tests running slow is when the test shows a bottleneck in my code, then a refactoring of the code is in order.

Share Improve this answer

answered Aug 13, 2008 at 23:54

Follow

Dale Ragan
18.3k ● 3 ● 55 ● 71

0

> How long is too long for an individual unit test?

I'd say it depends on the compile speed. One usually executes the tests at every compile. The objective of unit testing is not to slow down, but to bring a message "*nothing broken, go on*" (or "*something broke, STOP*").

I do not bother about test execution speed until this is something that starts to get annoying.

The danger is to stop running the tests because they're too slow.

> Finally, when you do decide the tests need to run faster, what techniques do you use to speed up your tests?

First thing to do is to manage to find out why they are too slow, and **wether the issue is in the unit tests or in the code under test ?**

I'd try to break the test suite into several logical parts, running only the part that is *supposedly* affected by the code I changed at every compile. I'd run the other suites less often, perhaps once a day, or when in doubt I could have broken something, and at least **before integrating**.

Share Improve this answer

Follow

edited Nov 24, 2008 at 8:40

answered Nov 20, 2008 at 12:35

philant

35.7k ● 11 ● 73 ● 113

0

Some frameworks provide automatic execution of specific unit tests based on heuristics such as last-modified time. For Ruby and Rails, AutoTest provides much faster and responsive execution of the tests -- when I save a Rails model `app/models/foo.rb`, the corresponding unit tests in `test/unit/foo_test.rb` get run.

I don't know if anything similar exists for other platforms, but it would make sense.

Share Improve this answer

Follow

answered Nov 24, 2008 at 8:47

Daniel Schierbeck

1,952 ● 2 ● 17 ● 24