Performance vs Readability

Asked 16 years, 4 months ago Modified 15 years, 1 month ago

Viewed 4k times

21

Reading this question I found this as (note the quotation marks) "code" to solve the problem (that's perl by the way).

```
100,{)..3%!'Fizz'*\5%!'Buzz'*+\or}%n*
```

Obviously this is an intellectual example without real (I hope to never see that in real code in my life) implications but, when you have to make the choice, when do you sacrifice code readability for performance? Do you apply just common sense, do you do it always as a last resort? What are your strategies?

Edit: I'm sorry, seeing the answers I might have expressed the question badly (English is not my native language). I don't mean performance vs readability only **after** you've written the code, I ask about before you write it as well. Sometimes you can foresee a performance improvement in the future by making some darker design or providing with some properties that will make your class darker. You may decide you will use multiple threads or just a single one because you expect the scalability that such threads may give you, even when that will make the code much more difficult to understand.

performance

Share

Improve this question

Follow

edited May 23, 2017 at 10:29

Community Bot

1 • 1

asked Aug 27, 2008 at 18:09

Jorge Córdoba

52.1k • 11 • 82 • 130

Good question. I too wonder about this when creating migration scripts that use perl regular expression substitution. It's a lot easier to maintain separate regular expression substitution expressions that gradually transform the input than it is to write one giant regular expression.
– Sridhar Sarnobat Oct 25, 2012 at 18:39

14 Answers

Sorted by: Highest score (default)

29

My process for situations where I think performance may be an issue:

1. Make it work.
2. Make it clear.
3. Test the performance.
4. If there are meaningful performance issues: refactor for speed.

Note that this does not apply to higher-level design decisions that are more difficult to change at a later stage.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:21

akmad

19.7k ● 2 ● 30 ● 25

8

I always start with the most readable version I can think of. If performance is a problem, I refactor. If the readable version makes it hard to generalize, I refactor.

The key is to have good tests so that refactoring is easy.

I view readability as the #1 most important issue in code, though working correctly is a close second.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:11

James A. Rosen

65.2k ● 62 ● 184 ● 263

7

Readability is most important. With modern computers, only the most intensive routines of the most demanding applications need to worry too much about performance.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:13

levand

8,480 ● 3 ● 42 ● 55

5

My favorite answer to this question is:

1. Make it work
2. Make it right
3. Make it fast

In the scope of things no one gives a crap about readability except the next unlucky fool that has to take care of your code. However, that being said... if you're serious about your art, and this is an art form, you will always strive to make your code the most per formant it can be while still being readable by others. My friend and mentor (who is a BADASS in every way) once graciously told me on a code-review that "the fool writes code only they can understand, the genius writes code that anyone can understand." I'm not sure where he got that from but it has stuck with me.

Reference

Share Improve this answer

Follow

answered Nov 5, 2009 at 21:04

Freddie

1,029 ● 10 ● 12

4

> Programs must be written for people to read, and only incidentally for machines to execute.
> — Abelson & Sussman, SICP

Well written programs are probably easier to profile and hence improve performance.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:18

Jared Updike

7,268 ● 8 ● 48 ● 73

3

You should always go for readability first. The shape of a system will typically evolve as you develop it, and the real performance bottlenecks will be unexpected. Only when you have the system running and can see real evidence - as provided by a profiler or other such tool - will the best way to optimise be revealed.

"If you're in a hurry, take the long way round."

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:16

Marcus Downing

10.1k ● 12 ● 65 ● 86

3

agree with all the above, but also:

when you decide that you want to optimize:

1. Fix algorithmic aspects before syntax (for example don't do lookups in large arrays)
2. Make sure that you prove that your change really did improve things, measure everything
3. Comment your optimization so the next guy seeing that function doesn't simplify it back to where you

started from

4. Can you precompute results or move the computation to where it can be done more effectively (like a db)

in effect, keep readability as long as you can - finding the obscure bug in optimized code is much harder and annoying than in the simple obvious code

Share Improve this answer

Follow

answered Sep 9, 2008 at 22:12

Oskar

2,294 ● 5 ● 28 ● 36

2

I apply common sense - this sort of thing is just one of the zillion trade-offs that engineering entails, and has few special characteristics that I can see.

But to be more specific, the overwhelming majority of people doing weird unreadable things in the name of performance are doing them prematurely and without measurement.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:15

Will Dean

39.5k ● 11 ● 92 ● 118

1

Choose readability over performance unless you can prove that you need the performance.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:12

17 of 26

27.4k ● 13 ● 68 ● 85

1

I would say that you should only sacrifice readability for performance if there's a proven performance problem that's significant. Of course "significant" is the catch there, and what's significant and what isn't should be specific to the code you're working on.

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:12

Jason Baker

198k ● 138 ● 382 ● 520

1

"Premature optimization is the root of all evil." - Donald Knuth

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:18

Chris Upchurch

15.5k ● 6 ● 52 ● 66

---

1 Partial and selective quotation. If you're going to quote it, quote the entire paragraph. You might be surprised by what it actually says. – user207421 Jan 31, 2016 at 23:54

Readability always wins. Always. Except when it doesn't. And that should be very rarely.

1

Share Improve this answer

Follow

answered Sep 9, 2008 at 22:16

rp.

17.6k ● 14 ● 65 ● 79

---

at times when optimization is necessary, i'd rather sacrifice compactness and keep the performance enhancement. perl obviously has some deep waters to plumb in search of the conciseness/performance ratio, but as cute as it is to write one-liners, the person who comes along to maintain your code (who in my experience, is usually myself 6 months later) might prefer something more in the expanded style, as documented here:

http://www.perl.com/pub/a/2004/01/16/regexps.html

0

Share Improve this answer

Follow

answered Aug 27, 2008 at 18:18

fool

There are exceptions to the premature optimization rule. For example, when accessing an image in memory, reading a pixel should not be an out-of-line function. And

0

when providing for custom operations on the image, never do it like this:

```
typedef Pixel PixelModifierFunction(Pixel);

void ModifyAllPixels(PixelModifierFunction);
```

Instead, let external functions access the pixels in memory, though it's uglier. Otherwise, you are sure to write slow code that you'll have to refactor later anyway, so you're doing extra work.

At least, that's true if you know you're going to deal with large images.

Share Improve this answer

Follow

answered Oct 9, 2008 at 19:10

Lev

6,657 ● 6 ● 30 ● 29