Why Neovim is Better than Vim

15 Jan 2015

I know Vim better than most. Vim was my first real text editor.[1] I used it for years. I helped write the Floobits plugin for Vim. I've delved into Vim's source code to figure out its workings. I even helped write a patch (though it was rejected). Considering these credentials, I hope you'll accept that I know what I'm talking about.

It may come as a shock when I say: The only good part of Vim is its user interface.

Every other aspect of Vim is irredeemable. The codebase is atrocious. The plugin API is cumbersome and restrictive. The dev community is apathetic. The benevolent dictator is averse to change. There is no chance of fixing these problems.

I wish it were otherwise, but it isn't.

Considering the degree of these criticisms, I should back them up with specific examples.

The Plugin API

Vim's plugin API is just plain bad. First, all plugin code runs synchronously. That means if any plugin's code is executing, Vim's UI is frozen. This makes many types of plugins difficult or impossible to implement. Linters have to finish in milliseconds or risk annoying the user. External commands (such as `make`) can't be cancelled, and they must finish before the user can resume editing.

Another annoyance is that writing plugins requires knowledge of Vim's special language: vimscript. This is true even if you're using a Vim compiled with support for other languages. Yes, `+python` gives you access to Python's libraries and syntax.

But your code will be littered with calls to `vim.eval()` and `vim.command()`. Here's an example:

```
import vim

# Show current directory in Vim
cwd = vim.eval('getcwd()')
vim.command(':Explore %s | redraw' % cwd)
```

You might notice that issues could arise from failing to properly escape variables in calls to `eval()` and `command()`. You'd be right. It's not uncommon for special character inputs to cause Vim plugins to crash or misbehave.

The Codebase

I started programming in C almost 20 years ago. Vim is, without question, the worst C codebase I have seen. Copy-pasted but subtly changed code abounds. Indentation is haphazard. Lines contain tabs mixed with spaces. Source files are *huge*. There are almost 25,000 lines in `eval.c`. That file contains over 500 `#ifdefs` and references globals defined in the 2,000 line `globals.h`.

Some of Vim's source code isn't even valid text. It's not ASCII or UTF-8. The venerable `file` can't figure out the encoding.

```
ggreer@carbon:~/code/vim% file -I src/digraph.c
src/digraph.c: text/x-c; charset=unknown-8bit
```

Thankfully, `eval.c` is pure ASCII.

Many of Vim's `#ifdefs` are for platforms that became irrelevant decades ago: BeOS, VMS, Amiga, Mac OS Classic, IRIX. These preprocessor statements may seem innocuous, but they slow development and inhibit new features. Also, Vim doesn't even work on most of these platforms anymore. It's just that nobody has an ancient system with which to test Vim. Neovim developers analyzed many of the preprocessor statements and found a significant number that could never be included in a working Vim.

Complexity stemming from cross-platform support may be excusable, but even something as simple as reading keyboard input is a nightmare in Vim. Stepping through with a debugger will result in call stacks such as `inchar()` in `getchar.c` calling `ui_inchar()` in `ui.c`, which calls `mch_inchar()` in `os_unix.c`, which calls `WaitForChar()`, which calls `RealWaitForChar()`. This call stack can be completely different on different platforms. It also differs when running in command line versus GUI mode.

Figuring out Vim's control flow is harrowing. Even when you hit paydirt in `RealWaitForChar()`, the code is extremely hard to follow. Here's a snippet. You can view the whole function at my Vim Hall of WTF.

```
# if defined(HAVE_GETTIMEOFDAY) && defined(HAVE_SYS_TIME_H)
    /* Remember at what time we started, so that we know how much longer we
     * should wait after being interrupted. */
#  define USE_START_TV
    struct timeval  start_tv;

    if (msec > 0 && (
#  ifdef FEAT_XCLIPBOARD
        xterm_Shell != (Widget)0
#   if defined(USE_XSMP) || defined(FEAT_MZSCHEME)
        ||
#   endif
#  endif
#  ifdef USE_XSMP
        xsmp_icefd != -1
#   ifdef FEAT_MZSCHEME
        ||
#   endif
#  endif
#  ifdef FEAT_MZSCHEME
    (mzthreads_allowed() && p_mzq > 0)
#  endif
        ))
    gettimeofday(&start_tv, NULL);
# endif
```

That `if` statement's conditions span 17 lines and 4 different `#ifdef`s. All to call `gettimeofday()`. Amusingly, even the body of that statement has a bug: times returned by `gettimeofday()` are not guaranteed to increase. User intervention or ntpd can cause the system clock to go back in time. The correct solution is to use a monotonically increasing time function, such Linux's `clock_gettime()` or OS X's `mach_absolute_time()`.

The Developer Community

Matt and I worked for months to add asynchronous functionality to Vim. From that experience, I have few good things to say about Vim's dev community. In fact, out of all the developer communities I've encountered, Vim's is the most hostile to change. Anything that isn't a bug fix is frowned upon.

Patches are often criticized for ridiculous reasons. After we posted our patch to the Vim-dev mailing list, the first reply was:

> NOTE: Don't use ANSI style function declarations. A few people still have to use a compiler that doesn't support it.

Seriously? C89 is a quarter-century old. The number of people stuck on older compilers can be counted on one hand. This is a non-concern. Still, I acquiesced. It was easier to make the change than argue with the critic.

The rest of that thread is me being as civil as possible, despite discouragement at every turn. The replies might as well be a paint-by-numbers guide on how to alienate new contributors.

On a more general note: After reading random posts on the Vim-dev mailing list, I get the impression that the developer community is fragmented. Some want Vim to be similar to Sublime Text: A flexible, extensible text editor for developers. Some (including BDFL Bram Moolenaar) are afraid of Vim becoming an IDE.

The Overly Cautious Dictator for Life

Speaking of Bram Moolenaar: His merge criteria are inscrutable. Some patches he ignores. Some, he attacks. Others, he merges.

Take a look again at the thread where Matt and I submitted our patch. We did our best to cater to Bram's every whim, but it was a waste of time. Had he immediately told us to give up, it would have been a better outcome for all involved. Instead, we were given hope and strung along, working on a patch that had no chance of getting merged.

The Alternative

A couple of months after my disillusionment with Vim, Thiago de Arruda submitted a similar patch. It was likewise rejected. But unlike me, Thiago didn't give up. He started NeoVim and created a Bountysource for it.

Neovim is exactly what it claims to be. It fixes every issue I have with Vim: The plugin API. The codebase. The community. The BDFL.

Neovim's plugin API is backwards-compatible with Vim, but it also allows asynchronous execution. Users have already made plugins that Vim can never have. For example, Neomake allows async linters. That feature alone is worth making the switch for.

Neovim's codebase is a substantial improvement. They've replaced much of the hacky, platform-specific code with libuv. They've fixed the problems with indentation, style, and bad file encodings. They've removed old code for ancient, unused platforms. They've drastically increased test quality and coverage. There's still much to be done, but the difference is already worlds better.

Neovim's development community is excellent. They respond to issues. They merge pull requests. They give quality feedback. And most importantly, they're nice to newbies.

In fact, they're nice to everyone. The main dev team holds no enmity toward Bram Moolenaar. They recognize Vim's failings, but they don't feel the need to criticize it.

The only thing Neovim is missing is a tagged stable release. But there's no need to wait. Right now you can clone Neovim, compile it, and have an editor that works with all your existing plugins.

If you are a Vim user, I strongly recommend switching to Neovim. It's the Vim you're used to, but with plugins you never knew you wanted.

Thanks to both Bjorn Tipling and Matt Kaniaris for their help with this post.

1. Edit and pico don't count.

← Burnout is in the Mind Optimizing Ag: Special-casing File Extensions →

When commenting, remember: Is it true? Is it necessary? Is it kind?

Comments for this thread are now closed ×

25 Comments **Geoff Greer's website** **Disqus' Privacy Policy** 1 **Login**

Recommend **26** Tweet Share **Sort by Best**

Thiago Arruda • 6 years ago • edited

Geoff, thanks for supporting Neovim like this.

I'd like mention a couple of extra reasons that IMO make Neovim better than Vim when looking it as an open source project instead of a simple set of features:

- UI organization with a lua based DSL created to improve functional testing. With this DSL it is possible to write tests that verify screen state and mouse/keyboard input in a way that is more readable than the current functional tests. Here's one example

- Tooling/CI: The community has done an amazing job with travis. We have automatic testing with valgrind/ASAN, documentation generation, dependency/nightly builds, and much more

IMO Better testing infrastructure is the most important, its what let us make aggressive changes to the code and be less conservative about receiving contributions.

53 ^ | ˅ • Share ›

Christian Brabandt • 6 years ago • edited

Well, this is my opinion as someone who has contributed several tens of patches to vim.

1) I do not remember any hostility on the vim_dev mailinglist.

2) If I remember correctly, your patch did not work on Windows and if I would be Bram, I wouldn't have included it, if it wouldn't work on all platforms.

3) ANSI style code: well the code is old and that is the style it was
when starting to develop vim. There is nothing wrong with keeping that
style.

4) Yes, some functions are a mess. But show me the open
source project, where all code is clean and nice and which has been
active for over 20 years.

5) I have been following the neovim development recently. There has been much activism, to remove working features that worked well and have therefore created problems for neovim, like the clipboard code or switching the Make program... This just sounds silly

6) I do not think, that removing the Windows functionalitiy will help you with respect to your user community.

7) I do not think, it is helpful, ranting here publically and trying to split the community between vim and neovim, rather try to be helpful and fix problems.

Sorry, I did not want to sound harsh, so if did, let me apologize...

26 ^ | ⌄ 8 • Share ›

Thiago Arruda ➔ Christian Brabandt • 6 years ago

> Yes, some functions are a mess. But show me the open
source project, where all code is clean and nice and which has been
active for over 20 years.

Lua is one such project.

> I have been following the neovim development recently. There has been much activism, to remove working features that worked well and have therefore created problems for neovim, like the clipboard code or switching the Make program... This just sounds silly

The old clipboard code was removed because it was huge and for the sake of maintainability, but that doesn't mean Neovim doesn't support clipboard: http://neovim.org/doc/user/...

Yes, some regressions are are introduced, but that is a natural consequence modifying such a fragile codebase. Even Vim is constantly merging bugfixes right? I bet many of these bugs are introduced by previous bugfixes which also happen to introduce regressions. Neovim is addressing this fragility by improving the test infrastructure, and eventually we'll break this vicious circle.

34 ^ | ⌄ 2 • Share ›

Christian Brabandt ➔ Thiago Arruda • 6 years ago

> I bet many of these bugs are introduced by previous bugfixes which also happen to introduce regressions.

So is Neovim. See the clipboard provider that has been causing more harm than good.

> Neovim is addressing this fragility by
improving the test infrastructure, and eventually we'll break this
vicious circle.

So does Vim. We have been constantly adding more and more tests, as we fix things. It's not like we do something without testing...

3 ^ | ⌄ 5 • Share ›

gosukiwi ➔ Christian Brabandt • 6 years ago

I can now get a taste of the Vim community, defending a mangled monster of spaghetti code. Refusing change. I love Vim but there's IMO no denial neovim is a change for good that had to be made if the project didn't want to end up as an unmaintainable beast. I guess there's people for everything :)

Here's a nice followup article on this: http://blog.aaronbieber.com...

20 ^ | ⌄ 5 • Share ›

dkgn ➜ Christian Brabandt • 6 years ago

> So does Vim. We have been constantly adding more and more tests, as we fix things. It's not like we do something without testing...

Still, new patches are not tested on the OSes/with the configurations Vim claims to support before they are merged. I think so, at least, otherwise I don't see how patches like https://code.google.com/p/v... or https://code.google.com/p/v... were required.

Neovim is also not there yet, but it already uses Travis to compile with different compilers under Linux and OS X, and will probably use AppVeyor to compile on Windows when that works again.

What I'm saying is that Vim may add more tests, but that doesn't improve the test infrastructure as such.

2 ^ | ˅ • Share ›

wal ➜ Christian Brabandt • 6 years ago

You said that you contributed several tens of patches to Vim. I don't really follow vim_dev in depth, but I think you're one of the top Vim contributors. Now consider Neovim: since it was started one year ago, it had over 1,000 closed pull requests. Sure, not all of that was merged, but at least 500+ contributions by all kinds of people went into Neovim. It's just really a magnitude easier to contribute to Neovim than it is to Vim. That is Neovim's big advantage.

People like Vim. Some of them don't think it's finished yet; they still see areas in which it can be vastly improved. Will this happen with Vim? Probably not (e.g. http://www.binpress.com/blo... "What does the future hold for Vim? Nothing spectacular. Mainly small improvements."). It will (and does) happen in Neovim (at least in my opinion).

Neovim attracts a lot of new contributors. Many of them have never looked at Vim's codebase. Things will break. But that's okay. People will get more experience with (Neo)vim's sources. From the interview I linked above: " How can the community ensure that the Vim project succeeds for the foreseeable future? Keep me alive." Neovim, in contrast, is not maintained by a single person. It's a community effort. That's its strength.

22 ^ | ˅ 1 • Share ›

Schell Scivally ➜ wal • 6 years ago

Though I wonder how many "merges" the vim code base has had since its creation 23 years ago. I don't know if we can tell, can we? Remember it predates svn by 9 years!

3 ^ | ˅ • Share ›

wal ➜ Schell Scivally • 6 years ago

Reading my comment (and yours) again, I think I should rephrase a bit. With this number, I didn't want to say that "Neovim has more PRs, therefore it's better". I wanted to say that many different people sent PRs, people who have never contributed to Vim development before. That might partially be because Neovim is a new project (and thus more "exciting"), but probably also because Neovim makes it easier to contribute, e.g. by using Github and continuous integration.

6 | 1 • Share ›

josh_at_dailydrip Schell Scivally • 6 years ago

This might be a great use case for Eric Raymond's new reposurgeon btw

1 | • Share ›

Amadeus Demarzi Christian Brabandt • 6 years ago • edited

Nailed it.

While I can completely understand the frustrations around wanting certain things in Vim (there's certainly a laundry list of things I would like that have not been implemented, and not even a realistic timeline for them to be implemented), but you get the sense there is some personal axe to grind by the author, and potentially other Floobits developers as well. I've been following nearly all the threads on this both on the mailing list and on Hacker News. Even the way this article was tweeted out by the author was sophomoric.

Bram has always been professional and courteous, perhaps he disagreed with the author's points of view, but he never attacked anyone. I think the same goes with the Vim dev mailing list, which I've been actively reading for the last couple of years now.

There is no doubt a tremendous amount of technical debt in Vim, although I know I am not qualified to speak for or against any of it.

If there is any major feedback that I would have for Vim development, it would be that there should be both a bleeding edge/development version, and a stable version. I think many of the changes that go into vim tend to be overly cautious with little room for mistakes since the only branch is the release branch. I believe there should be an opportunity for things to go in that could potentially break things, and give time for the active community to suss out the issues and get them fixed.

8 | • Share ›

Shougo Amadeus Demarzi • 6 years ago

> If there is any major feedback that I would have for Vim development, it would be that there should be both a bleeding edge/development version, and a stable version. I think many of the changes that go into vim tend to be overly cautious with little room for mistakes since the only branch is the release branch. I believe there should be an opportunity for things to go in that could potentially break things, and give time for the active community to suss out the issues and get them fixed.

Yes. You are right. But it is impossible.

Because, vim is single developper product.

To maintain stable branches, stable branch maintainer is needed like Ruby.

6 | • Share ›

splbio • 6 years ago

This is awesome to hear! Good luck guys, I'll be giving newvim a try asap. I may even add a FreeBSD package for it (if one doesn't already exist).

4 ^ | ⌄ • Share ›

mattn • 6 years ago • edited

What mean Better? NeoVim is removing compatibility, Windows CUI or etc.

4 ^ | ⌄ • Share ›

Thiago Arruda ➦ mattn • 6 years ago

Maybe it isn't better from the compatiblity POV yet, but that will change.

When Neovim started I removed a lot of old platform support, but it wasn't because I'm against writing cross-platform code, I just think the way Vim does it is wrong and not sustainable: Gigantic functions with thousands of LOC and mixed #ifdefs that few people can understand.

As the refactoring progresses, Neovim code will be better organized and it will be possible to add a lot of lost platform compatiblity back. My goal is to make Neovim core very close to pure C99, and split IO/OS stuff into distinct layers that will make platform-specific code more easy to maintain. In a distant future, Neovim will be a C99 library and even the libuv/async code should be replaceable.

As for windows support, I have began a big refactoring of the UI layer that removes a lot of the unix-specific code from the core and hopefully will get Neovim much closer to a windows build. This refactoring enabled a fully working Gtk UI to be implemented in pure python that theoretically should work on windows when connected via msgpack-rpc to a Neovim instance running on unix.

27 ^ | ⌄ • Share ›

gosukiwi ➦ Thiago Arruda • 6 years ago

I'm kind of new to Vim. Been using it for a few years and I'm a young web developer. I've learnt to love the vim way of doing things and by now I find it impossible to live without Vim.

If I'm forced to use Eclipse or Visual Studio, I MUST install a vim-mode plugin, same for Sublime Text or Atom. I'm really grateful for your work on Neovim, seems like it's a step in the right direction which finally someone had the cojones to do :)

Thank you and all of neovim comunity for this, even though I work mostly on Windows, I know eventually I'll be able to use a new, improved Vim (ha, improved vi improved!). It's really great to see how much the source code has improved already! Keep up the good work!

11 ^ | ⌄ • Share ›

Stefano Borini • 6 years ago • edited

I personally gave up on both, and started writing my own version, vai (see https://github.com/stefanob...), fully in python. I'd love to elaborate more, but I cannot add anything more than what I already said in the RATIONALE (see https://github.com/stefanob...) document. Unfortunately, I am alone...

3 ^ | ˅ 2 • Share ›

gosukiwi ➦ Stefano Borini • 6 years ago • edited

I've also tried to replace Vim. But unfortunately nothing comes close enough, a bunch of "vim plugin" for the most popular editors lack some features I use, like block selection. I also love beeing able to swap to NerdTree using just my keyboard, and swapping files the same way (FuzzyFinder). If I use a plugin I still have to use the mouse to select files / tabs, which slows me down and mostly annoys me.

Why Python though? Just wondering :P

^ | ˅ • Share ›

Juan Enrique Munoz Zolotoochin ➦ Stefano Borini • 6 years ago

I've tried many editors, looking for a replacement for vim. Most of them looked promising but they lacked the single feature that would make me switch: existing vim plugin compatibility.

I just briefly read your readme at github and I didnt see this mentioned. But if you want to build a community you need users, so you need to solve the chicken and egg problem: http://www.joelonsoftware.c...

Neovim seems to have solved it, so Im definitely going to try it.

^ | ˅ • Share ›

Stefano Borini ➦ Juan Enrique Munoz Zolotoochin • 6 years ago • edited

> I just briefly read your readme at github and I didnt see this mentioned

I have no interest in keeping compatibility for vim plugins, nor I need at the moment a plugin community. I just need to develop a good environment, and plugins will follow. neovim takes the wrong route of keeping vim compatibility and vim language, in addition to keep writing in C. The Lua environment is promising, but is it going to keep interest? who knows lua, except for WoW modders? In my opinion, it's never going to have the flexibility needed for fast development to create more than an editor. I don't want an editor. I want an IDE, with a terminal interface.

Besides, neovim hasn't released anything yet, so it's hard to speculate, but considering the amount of time and investment they have, and comparing to what I obtained just by myself, I would have a much better editor if I had that kind of resources.

2 ^ | ˅ 2 • Share ›

dkgn ➦ Stefano Borini • 6 years ago

> neovim takes the wrong route of keeping vim compatibility and vim language

Drop all that and cut off the whole Vim community? I don't think that would've been a good move...

Neovim has support for many languages, not just Lua. Python is officially supported (there's even a GTK UI written in Python), there are other unofficial plugins for Go, Ruby, Julia, ... it's quite easy to add support for a new language, as Neovim's "core" doesn't need to be modified for that.

While Neovim didn't release anything yet, I used in on my system for several months, and it works quite well.

4 ^ | ˅ • Share ›

Stefano Borini ➜ dkgn • 6 years ago

I don't care, I keep working on mine.

^ | ˅ 5 • Share ›

Guest • 6 years ago

Geoff, thanks for supporting Neovim like this.

I'd like mention a couple of extra reasons that IMO make Neovim better than Vim when looking at it as as an open source project: