On Linux kernel maintainer scalability

By **Jonathan Corbet**
October 12, 2016

LinuxCon Europe

LWN's traditional development statistics article for the 4.6 development cycle ended with a statement that the process was running smoothly and that there were no process scalability issues in sight. Wolfram Sang started his 2016 LinuxCon Europe talk by taking issue with that claim. He thinks that there are indeed scalability problems in the kernel's development process. A look at his argument is of interest, especially when contrasted with another recent talk on maintainer scalability.

Beyond changesets merged

Sang's core point is that looking at the number of patches merged only tells part of the story; it says nothing about what had to happen to get those patches into the mainline. Looking at the last few years' worth of development cycles, he noted that relatively few patches carry tags beyond the Signed-off-by applied by the developer and the committer. In particular, around the 3.0 days, only about 20% of the patches in the mainline had an Acked-by, Reviewed-by, or Tested-by tag indicating that anybody other than the maintainer had seriously looked at them. That number is closer to 40% in current kernels, he said; it is a clear improvement, but still does not make him happy. For a properly scalable kernel process, he said, we should have much higher levels of review by developers who are not the subsystem maintainer.

Another metric one can look at is the time difference between the date on the patch and the date on which it was first committed to a git tree. The Ethernet driver maintainers, he said, are heroes: 80% of all the patches were accepted within two weeks. A number of other subsystems do not do anywhere near as well, and some have gotten significantly worse. I2C, Sang's own subsystem, has stayed about the same over the last three years, which surprised him. As the workload has increased, it has come to feel like things are getting much worse.

The time-to-commit metric may be useful, but it is not without its flaws. The final version of a patch may have been committed fairly quickly, but previous versions could have languished without review for a long time. Patches that are rejected or that get lost are not considered at all.

One way to try to get a better handle on things is to look at the Patchwork systems for the subystems that use it, and, in particular, to look at the backlog of patches found there. For I2C, it shows a relatively low backlog until about 3.16, when he gave up on trying to keep up with the flow and fell behind. The ACPI

subsystem has an amazing backlog of zero. The relevant maintainer (Rafael Wysocki) was in the room; he noted that it depends on how a subsystem uses Patchwork. He said that he quickly marks a lot of patches as inapplicable; Sang replied that he doesn't even have the time to do that. The ext4 filesystem shows a linear growth in its backlog, up to about 800 patches currently. The numbers for several other subsystems were shown; almost all of them are going up.

The problem, Sang said, is that the number of committers is not scaling to match the growing number of contributors to the kernel. We are getting more reviewers, but they are coming in slowly and are not anywhere near enough. As a result, the number of unprocessed patches is on the increase.

How can this problem be addressed? Users can help by commenting on and, especially, testing patches. Developers need to be aware that sloppiness is often a problem; they should acknowledge when they have done suboptimal work. Developers need to take part in reviewing; if nothing else, they should review their own patches. For maintainers, working harder is not generally the solution; that just leads to burnout. They should get their tools in order and automate tasks whenever possible; looking at what other maintainers are using can be helpful. Companies should allow and encourage their developers to spend time reviewing patches.

What he does not want to see is a "kernel infrastructure initiative". The Core Infrastructure Initiative, run by the Linux Foundation as a way to channel resources to important but underfunded projects, is a good thing, but it is a reaction to a problem that got out of control. Things had to go wrong first. Sang would rather see action now to keep things from getting to that state.

For I2C, Sang intends to step back a bit. He will become one of the I2C developers, one of its architects, and one of its reviewers, but he will not be the only one. That may slow things down in the short term, since he will be doing less patch review. The advantage is that he will stay sane, and will have the time and energy to try to address the problem on higher levels.

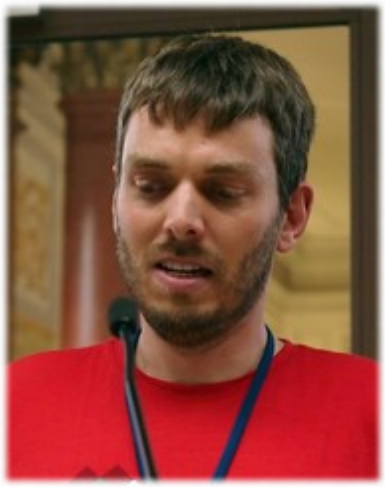
The maintainer as bottleneck

While Sang intends to step back on patch review, his plan still calls for him to be the sole committer of patches for the I2C subsystem. In this context, it is interesting to look at another talk, given at Kernel Recipes one week earlier by i915 graphics driver maintainer Daniel Vetter. He, too, made the point that maintainers don't scale, but he would rather see maintainers get help at all levels.

One year ago, he would have said that there was no problem in the i915 subsystem. Applying patches was relatively easy, after all. He had never reviewed the majority of the patches there; i915 has a number of developers who can do that. But, as the single maintainer, he gave the subsystem "a bus factor of one"; when he wasn't available for any reason, things simply came to a stop.

At the 2015 Kernel Summit, Linus Torvalds said that he has come to like the group maintainer model, where more than one person takes responsibility for a given subsystem. Vetter wanted to give that a try, but he quickly ran into a problem: nobody was willing to sign up as the co-maintainer for the i915 subsystem. He was, however, able to find developers who were willing to commit patches for i915; indeed, he signed up 15 of them. He figured he would experiment with the multiple-committer model for one release cycle. After all, nobody had ever really tried this before in the kernel, so it must be a stupid idea.

That was one year ago, he said, and disaster has failed to materialize. Instead, he has "seriously happy contributors," and a whole set of reviewers who can apply the patches they look at. He is now "a bored maintainer," and all of the nagging and begging to get code



merged has gone away. He has found that commit rights are a strong carrot that can be used to get developers and companies to contribute — and to be careful about the work they do. It also leads to "distributed conflict management" that makes life easier.

So what does he do anymore? His main job at this point, as "the" maintainer for i915, is communications with the outside, including any work that requires coordination with other subsystem trees. He connects developers with the appropriate reviewers, and puts together the pull requests to send work upstream. And, of course, he "takes the blame for everything".

To make this model work, he said, a subsystem clearly needs a team of developers, and non-maintainer reviews must be the norm. The group should be consistent, with developers who stay around; otherwise, enforcement via social feedback will not work well. Good documentation and tools are necessary; i915 has a set of process documents on this page. When somebody makes a mistake, if possible, a check should be put into the tools to keep it from happening again.

Good testing is crucial to this model. A multi-committer tree can never be rebased, so there is no way to remove embarrassing mistakes. They really need to be avoided in the first place; that requires good pre-commit testing to ensure that the obscure corner cases do not break.

The rough consensus model works best for a group like this. The default on any patch is "no action", so a developer's full disagreement will stop things. What's important, he said, is to have agreement on the goals for the subsystem; disagreement on the path taken toward those goals is acceptable. A good rule of thumb is "if you push a patch and there's screaming on IRC, you shouldn't have done it."

In general, he said, the kernel could probably benefit from more maintainer groups like this. It is a more efficient way to maintain busy subsystems, especially those that currently have a lot of submaintainer trees.

Meanwhile in Berlin

Fast-forward one week; your editor raised this idea in Sang's talk and asked whether the single-committer model might be part of the scalability problems raised there. The developers in that room tended toward skepticism over whether the idea could work outside of the i915 tree. Wysocki, in particular, seemed to feel that there were relatively few submaintainers who could be trusted with full commit access. These maintainers push patches that must be rejected fairly often, so they should not be able to commit directly to the subsystem tree.

Perhaps these developers, too, would be pleasantly surprised if they were to run an experiment with more widely distributed commit rights. In any case, it seems likely that growing numbers of developers and patches will put more stress on subsystem maintainers. If those maintainers are not to become a choke point for kernel development, ways to spread the work they do will be required.

[Your editor thanks both the Linux Foundation and Kernel Recipes for supporting his travel to these events.]

On Linux kernel maintainer scalability
Posted Oct 13, 2016 8:38 UTC (Thu) by **vegard** (subscriber, #52330) [Link]

Regarding reviews and patch quality, this is an old email I have from Andrew Morton on the topic:

"""
Date: Wed, 16 Jul 2008 17:10:52 -0700
From: Andrew Morton <akpm@linux-foundation.org>
To: "Vegard Nossum" <vegard.nossum@gmail.com>
Cc: penberg@cs.helsinki.fi, torvalds@linux-foundation.org, mingo@elte.hu
Subject: Re: [git pull] RCU updates for v2.6.27

On Thu, 17 Jul 2008 01:47:05 +0200
"Vegard Nossum" <vegard.nossum@gmail.com> wrote:

> How about starting to reject patches which have no reviewed-by tag?

Yeah, this is sorely tempting.

At kernel-summit-07 I discussed and basically proposed the Reviewed-by:
thing and everyone was surprisingly agreeable. I think I floated the
idea of making it obligatory but in no way have I pushed it at all. I
just don't have the energy and the grief-absorption-capability.

It would be a huuuuuge change. A dramatic one. It would quickly cause
the organisation of an economy in which people find they need to
"trade" reviewing activity. People would understand that if they don't
review other people's stuff then others won't review their stuff and
they don't get their patches merged.

Would it increase kernel quality? Yes, quite a bit I expect.

Would it slow things down? Yes, it would.

But both of those are the same thing.

Should we do it? Well, yes, I think we should. There would be some
bumps and glitches but once we got into the swing of it, the impact
would be not too bad. After all, it is just a formalisation of
something which we're supposed to be doing already! If we were doing
that which we _know_ we should be doing, the impact would be zero.

can we do it? Well, the steps are:

1: gather general consensus that we have some problem

2: get agreement and that the problem needs fixing

3: get agreement that this process step is one way of fixing said problem

4: implement it.

I think that we'd have a good chance of succeeding in all these steps.
After all, the winning debating point here is that *we're already
supposed to be reviewing all patches*.

Problem is, do I have the requisite energy to try to inflict this on
everyone at KS08? It's looking doubtful at present. It's much easier
if Linus just stands up and says "you all suck and we're doing this".
"""

I really do think reviews are undervalued in the sense that the reviewer often don't get very much out of it for themselves. You can put hours into a review but that is not recognised beyond a single "Reviewed-by" line in the final commit; in terms of personal exposure (which I do believe is a big factor for non-maintainer contributors, for better or worse), it is much more attractive to write code and patches.

Maybe maintainers should include review and reviewer stats in their pull requests, crediting not just the patch authors, but also the reviewers. This would also make it more obvious when large batches of patches have not been reviewed.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 13, 2016 8:41 UTC (Thu) by **vegard** (subscriber, #52330) [Link]

> You can put hours into a review but that is not recognised beyond a single "Reviewed-by" line in the final commit; in terms of personal exposure (which I do believe is a big factor for non-maintainer contributors, for better or worse), it is much more attractive to write code and patches.

(Not to mention commits which have already been made. I occasionally look over a pull request when I see it on LKML, but by that time there is no way to record the review in git history.)

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 17, 2016 14:51 UTC (Mon) by **imMute** (subscriber, #96323) [Link]

Maybe code reviews should be marked using tags instead of commit message strings – that way they can be added after-the-fact.
I suspect Git's refspec will need to learn to "exclude" things rather than being inclusive only, otherwise people who pull tags will get overloaded with them.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 13, 2016 9:58 UTC (Thu) by **blackwood** (subscriber, #44174) [Link]

I wasn't aware of this mail from Andrew, but this is what I've done in i915, and essentially what's going on in the drm subsystem at large. Reviews are required, period. And yes we have a flourishing tit-for-tat review market going on, coordinated mostly over irc ;-)

One key bit to make that happen imo is that the maintainer (even with committer model) is a great example, and goes to great pains to get a review even for trivial patches, if those patches are their own. Otherwise everyone stops bothering with haggling for some reviewer's bandwidth real fast.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 18, 2016 9:02 UTC (Tue) by **wsa** (guest, #52415) [Link]

Thank you for pointing out Andrew's mail. I couldn't agree more. What I forgot to mention in my talks: I2C will require "Reviewed-by" tags in the near future as well. Let's see how that works. The idea of mentioning reviewers in pull-requests is great, I'll definately do that!

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 13, 2016 8:46 UTC (Thu) by **blackwood** (subscriber, #44174) [Link]

When discussing scalability with other maintainers I hear the complaint that there's lack of trustworthy submaintainers, or reviewers or whatever fairly often. Like I said in my talk, I never really had the problem, neither with reviewers (that worked well from the start and constantly grew to more people) and submaintainers (candidates refused to sign up, not a problem with lack of them). I think a primary role as maintainer is to grow people into these roles and make sure there's enough, at least in a busy subsystem with plenty of contributors. If you fail to do that, then you're not a great maintainer in my opinion.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 13, 2016 16:51 UTC (Thu) by **wsa** (guest, #52415) [Link]

Subsystems differ. As tglx nicely put it at LinuxCon, I2C is more of a "drive-by" subsystem. You do the driver, get it accepted, then move on to, say, SPI. This is different from GFX and network drivers which need more constant attention. So IMO, there it is easier to keep developers also attached to the whole subsystem, too.

That being said, I got co-maintainers for ACPI and muxes. I definately am willing to share responsibility and do that when I see talent. Currently, though, there are exactly 0 regular reviewers on the list which I could award with extra responsibility.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 24, 2016 14:44 UTC (Mon) by **jengelh** (guest, #33263) [Link]

It also depends on the subsystem. Networking and filesystems are easy, you can test whatever you have on almost any machine, not dependent on particular hardware.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 14, 2016 0:18 UTC (Fri) by **rjw@sisk.pl** (subscriber, #39252) [Link]

> When discussing scalability with other maintainers I hear the complaint that there's lack of trustworthy submaintainers, or reviewers or
> whatever fairly often. Like I said in my talk, I never really had the problem, neither with reviewers (that worked well from the start and
> constantly grew to more people) and submaintainers (candidates refused to sign up, not a problem with lack of them).

That merely means that your experience is different from the experience of the others.

> I think a primary role as maintainer is to grow people into these roles and make sure there's enough, at least in a busy subsystem
> with plenty of contributors. If you fail to do that, then you're not a great maintainer in my opinion.

Well, thanks. I felt I was doing something wrong. :-)

Maybe not everyone is like you and not every piece of the kernel code they maintain is like yours.

There are some good (and responsive) reviewers for some pieces of code I maintain and for some of them I only pick up patches reviewed by certain people (and some of them are listed as maintainers of those pieces of code even), but I do commit the majority of patches and it looks like this works for everybody involved. I have no problems with that either.

I have to admit that I wouldn't have been able to handle the volume without those reviewers, but then my experience with people sending me pull requests and so on is seriously less than fantastic.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 13, 2016 15:38 UTC (Thu) by **fratti** (guest, #105722) [Link]

It's interesting to see that the "Linus doesn't scale" problem has moved further down the maintainer tree as kernel development accelerates.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 14, 2016 18:23 UTC (Fri) by **jani** (subscriber, #74547) [Link]

To be fair, i915 hasn't relied on a single maintainer for nearly three years now. Going from 1 to 2 maintainers was the hard part, and once we'd gotten all the tooling and documentation together for that, going from 2 to 15 wasn't much of a bump in the road.

Reply to this comment

On Linux kernel maintainer scalability
Posted Oct 17, 2016 12:47 UTC (Mon) by **blackwood** (subscriber, #44174) [Link]

Yeah I pretty badly screwed up the opening for my talk here. Jani's been an awesome co-maintainer, and he's doing a stellar job wrestling -fixes, bug reports and a lot of related seemingly boring but rather important things. And of course polishing all the overall process issues. But since we specialize quite a lot we'd still have a rather big problem if either of us disappears, and that's what I meant to express with a essentially a bus factor of one. But with stage panic and all that I made a mess of it all and dropped all the important bits :(

Reply to this comment