Vetter: Linux Kernel Maintainer Statistics

[Posted April 23, 2018 by corbet]

Daniel Vetter looks at some kernel-development statistics, with a focus on patches written by the maintainers who commit them. "Naively extrapolating the relative trend predicts that around the year 2025 large numbers of kernel maintainers will do nothing else than be the bottleneck, preventing everyone else from getting their work merged and not contributing anything of their own. The kernel community imploding under its own bureaucratic weight being the likely outcome of that. This is a huge contrast to the 'everything is getting better, bigger, and the kernel community is very healthy' fanfare touted at keynotes and the yearly kernel report. In my opinion, the kernel community is very much not looking like it is coping with its growth well and an overall healthy community."

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 5:23 UTC (Tue) by **pbonzini** (subscriber, #60935) [Link]

There are some points that I disagree with, but this is much more constructive than Daniel's talk, even with the gloomy forecasts.

For example:

1) Among the high-number-of-unreviewed-maintainer-commits there's net, which is a very hierarchical subsystem IIRC. So even if you don't see a Reviewed-by or Signed-off-by tag, you have to check whether the maintainer next up in the chain is looking at what he's merging. I know Daniel prefers group maintainership to hierarchical maintainership (and giving commit rights to receiving pull requests), but hierarchical maintainership is a thing too. Likewise, for the tip tree, is the person writing the commits the same that is sending the pull request to Linus?

2) What percentage of the maintainer self-commits are trivial bugfixes? That could be easily parsed from a Fixes tag and would be the difference between

> This indicates that the kernel community fails to mentor and train new maintainers at a pace sufficient to keep up with growth.

This indicates that the kernel community is successful in making developers stick to the project, so that they can fix their own mistakes instead of having the maintainer having to step up and do that.

3) Saying that "Current maintainers are ever more overloaded, leaving ever less time for them to write patches of their own and get them merged" presumes that maintainers _have to_ write patches. Working for others can be just as rewarding, even if you write less code (or if your code ends up outside the kernel, for example in a repository of tests).

It's interesting that KVM ends up in the good list in Daniel's post, and I'm curious whether that should make me more sympathetical to his points, or rather blind to the actual problems just because I don't see them. Certainly I would like to talk in person.

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 5:27 UTC (Tue) by **pbonzini** (subscriber, #60935) [Link]

Just to be clear, I'm writing the above objections hoping that we can get an even better picture of kernel development, not because I think Daniel's post is bad!

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 5:56 UTC (Tue) by **blackwood** (subscriber, #44174) [Link]

Some thoughts on these questions:

1) The issue is (not just talking about net) that there's both a lot of such commits (in absolute numbers) and fairly few (relative to the overall pull, at least for most pull requests. You really need to go looking for them using scripts (we do that in our group maintainer scaffolding used in drm), or you'll miss them. The other issue is more practical: Do you really want to reject a pull with 100 patches because 1 maintainer self-commit wasn't up to standards? From my experience and chatting with lots of other people, that just doesn't happen. It's plenty easier to do if you have lots of small topic pull requests, and I guess some of them get dropped/delayed each release cycle. But I didn't find any indication of review happening there either, and imo if it's not worth recording, how much valued is it really?

So yes, in theory, the maintainer above should be checking this. In practice I don't think that happens all that often.

2) I did that analysis for the drm 4.17 merge window pull manually. There's no way to script this and filter out the oopsies/trivial bugfixes. Fixes: Could easily be a trivial compile fix for a 0day randconfig, or a really tricky regression fix that really wants to see some review before being merged.

And even for trivial patches there's still the double-standard, since mere contributors can't get their own trivial patches merged without the implied maintainer-ack. In graphics at least the drm-intel and drm-misc trees also require acks for trivial stuff.

3) I covered this in the article already: Self commits going down as a maintainer becomes more senior is expected, for exactly the reasons you lay out. But there should be a supply of more junior maintainers (who will focus more still on their own work, instead of more on helping others), and so in aggregate I expect the number of self-commits to stay (relative to overall commits) the same. But it's going down.

I'd like to explore a bit more where those maintainers have gone to, maybe by looking at some other subsystem (arm-soc looks interesting, there's at least lots of maintainers there). And also looking at how many people do self-commits each release and how that develops over time. But the article is already long as-is, so next time around.

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 11:05 UTC (Tue) by **pbonzini** (subscriber, #60935) [Link]

This is getting interesting. Again I don't want to diminish the value of your analysis, just trying to understand what may be missing. As you note, it's hard to extract the ins and outs of a community just by scraping git logs.

> But I didn't find any indication of review happening there either, and imo if it's not worth recording, how much valued is it really?

Maybe an interesting thing to analyze is (for subsystems that have sub-pull-requests) the average size of a sub-pull-request. For smaller pull requests, review can happen but it isn't recorded in the same way as patches that have a Signed-off-by without a Reviewed-by. The implicit review here is recorded in the merge commit.

Another possibility is that what you see as a single monolithic subsystem is actually a collection of small subsystems, with a single shepherd between them and Linus. They could still benefit from group maintainership and a more explicit ack/review process, but the picture would be very different from that of a core subsystem that throws code into linux.git without any review. I would guess that the net statistics would be very different between net/ and drivers/net/, for example.

> And even for trivial patches there's still the double-standard, since mere contributors can't get their own trivial patches merged without the implied maintainer-ack. In graphics at least the drm-intel and drm-misc trees also require acks for trivial stuff.

For KVM, "silence gives consent". If I post trivial patches and push them to my (public) staging git branch, they remain there until someone complains or a few days have passed. I do Cc the people that may be interested in a review. So another interesting metric is, are any people Cced on unreviewed maintainer self-commits?

I would like to understand why you have such a negative opinion of "the guy who actually commits patches to the tree". To me, that's just one of the things that one can do to help running the community. Yes, sometimes the wrong person is assigned that task, but is it correct to turn that into a case against the task itself?

> But there should be a supply of more junior maintainers (who will focus more still on their own work, instead of more on helping others), and so in aggregate I expect the number of self-commits to stay (relative to overall commits) the same. But it's going down.

Why would those people _have_ to be maintainers? Even though I like group maintainership, one also has to recognize that more maintainers bring more overhead. I find it just as likely that newcomers end up focusing on a particular file or feature and gradually become the expert on that (and the maintainers actively seeks their opinion!), or that a company donates testing infrastructure. This distributes the maintainers' job and creates a community around the task of reviewing patches.

And if you do have more maintainers, sending pull requests to each other, then you might end up in the situation of bullet 1 where the act of reviewing happens but isn't recorded except through a merge

commit...

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 13:50 UTC (Tue) by **blackwood** (subscriber, #44174) [Link]

> Maybe an interesting thing to analyze is (for subsystems that have sub-pull-requests) the average size of a sub-pull-request. For smaller pull requests, review can happen but it isn't recorded in the same way as patches that have a Signed-off-by without a Reviewed-by. The implicit review here is recorded in the merge commit.

DRM is definitely such a subsystem, with a bunch of the smaller driver trees being mostly not reviewed self-commits. Beyond an overall "does it compile" assessment there's not really any detailed (and definitely no per-patch) review going on.

> Another possibility is that what you see as a single monolithic subsystem is actually a collection of small subsystems, with a single shepherd between them and Linus. They could still benefit from group maintainership and a more explicit ack/review process, but the picture would be very different from that of a core subsystem that throws code into linux.git without any review. I would guess that the net statistics would be very different between net/ and drivers/net/, for example.

Yeah, splitting up net int drivers and core networking would be interesting. The trouble is that comparatively few things are maintained in sub-trees in the networking world (more than 50% goes in through Dave's tree directly). So you'd need to do a combination of git tree and path based analysis to figure out what's going on. That get's rather hairy.

I also didn't want to dig into other people's subsystems, just kernel overall and graphics.

> I would like to understand why you have such a negative opinion of "the guy who actually commits patches to the tree". To me, that's just one of the things that one can do to help running the community. Yes, sometimes the wrong person is assigned that task, but is it correct to turn that into a case against the task itself?

The linux kernel having relatively few committers compared to other open source projects isn't what I'm critizing here. I think it's not the most efficient way to run things (imo graphics would be real pain if we'd have half as many committers/maintainers as we do), but alas.

What irks me with self-commits that don't have a record of oversight by a 2nd person is that it creates a double-standard: Maintainers can get away with avoiding review, normal contributors (and they make 80-90% of contributions in a normal kernel subsystem, kvm being one of the exceptions here) simply cannot avoid that loop to jump through, even after having been around for years. In my opinion, the same rules should apply for everyone.

> > But there should be a supply of more junior maintainers (who will focus more still on their own work, instead of more on helping others), and so in aggregate I expect the number of self-commits to stay (relative to overall commits) the same. But it's going down.

> Why would those people _have_ to be maintainers? Even though I like group maintainership, one

> also has to recognize that more maintainers bring more overhead. I find it just as likely that newcomers end up focusing on a particular file or feature and gradually become the expert on that (and the maintainers actively seeks their opinion!), or that a company donates testing infrastructure. This distributes the maintainers' job and creates a community around the task of reviewing patches.
>
> > And if you do have more maintainers, sending pull requests to each other, then you might end up in the situation of bullet 1 where the act of reviewing happens but isn't recorded except through a merge commit...
>
> This is all valid, and it does explain how the kernel community manages to merge all that stuff with rather few people actually committing things. I think it's not the best way to organize a subsystem - group maintainership, even with fairly big group works much better than lots of trees with pull requests, or lots of reviewers with just a few patch monkeys doing all the applying.
>
> What I'm worried about is that it's going down, while the kernel is growing. And as shown with the gpu subsystem, it's possible to keep up with the growth of an area and keeping the maintainer ratio (whatever you think is the optimum, or just how that subsystem traditionally has run the show) steady.
>
> I'm also clear that counting self-commits is a pretty lousy metric to measure "how many maintainers do we have". But it's one you can measure and see historical trends. Perhaps looking at s-o-b chains would be interesting too, but at least in the gpu subsystems those generally happen for internal trees (for new hw that's not yet publicly announced), or from vendor trees. Both not really indicative of upstream maintainers. The big exception is mmotm, but I'm handling that already.
>
> I also don't know why this is happening (I'm playing around with new ways to look at this, thus far nothing). But through the best available metric I could figure out it looks like maintainers overall are not keeping up with the growth. Aside: I didn't expect this when compiling the data at all. I knew it was fairly low, but assumed it'd be more or less steady. Everything else in the data was more or less matching my expectations.

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 14:08 UTC (Tue) by **blackwood** (subscriber, #44174) [Link]

Maybe some of the ways I looked into measuring "how many maintainers do we have" that don't really work:

- Counting entries in MAINTAINERS. We have plenty people even in the gpu subsystem that submit their one driver, getting their maintainer entry added and then disappear. So just counting maintainer entries doesn't tell you who's actually active (and how much they're doing).

- Counting commits of people listed in MAINTAINERS. We have a big chunk of folks who are maintainer for something, but mostly write fixup patches all over the tree (or heavily contribute in other areas). I think given how hierarchical and big the kernel community is, counting all commits someone does even in areas they're not maintainers doesn't tell us much. It also makes things really complicated if you want to look at a specific subsystem or git tree.

> The inverse also happens: Lots of people assume maintainer duties (of last resort, or just temporary, or whatever) for areas they're not listed in MAINTAINERS.
>
> - Counting reviews (whether listed in MAINTAINERS or not). Reviewed-by tags in general are on an uptick (a slow one, but it's fairly steady). So you'd need to do some pretty serious statistic to try to compute out the overall trend to see how many maintainers there are, and I don't think any kind of useful signal will come out of that exercise.
>
> - Looking at replies on the mailing lists. That would be a full-time research project, starting with simply getting reliable archieves of all the mailing lists used by the kernel community. And then trying to match patches against their discussions (usually the diffs don't match, and often the patch id doesn't either due to fixups when applying).
>
> Given all that I figured looking at self-commits is the best metric I can have: I can crunch the numbers on the side without this becoming a full time research project. And it since it looks at what actually happens instead of what's documented in MAINTAINERS it has a higher chance of reflecting reality.

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 27, 2018 21:24 UTC (Fri) by **bfields** (subscriber, #19510) [Link]

"So even if you don't see a Reviewed-by or Signed-off-by tag, you have to check whether the maintainer next up in the chain is looking at what he's merging."

I pretty much assume that pulls are exactly for the case where you're delegating review to someone else.

If I'm willing to review individual patches, I may as well apply them while I'm at it.

As I see it the advantages of a pull/merge over applying a patch series is a) it's a quicker more convenient way to merge a whole bunch of development, and b) it records *exactly* the state of the other developer's tree, so that if a problem is introduced we'll know whether it came from my tree, their tree, or the merge itself.

Both of those are most useful if I'm delegating the review to someone else rather than doing it myself.

So I'd take "X pulled Z from Y" to be a pretty strong hint that X did *not* review Y. (But rather than X trusts Y to do a good job of Z.)

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 6:41 UTC (Tue) by **sandeen** (guest, #42852) [Link]

"What would Dave Chinner do?"
While that's not a bad mantra, I'll point out that the very capable Darrick Wong is the current XFS kernelspace maintainer. The XFS developer community had a successful handoff from one maintainer to another, and retained the 100% review standard.

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 24, 2018 7:04 UTC (Tue) by **blackwood** (subscriber, #44174) [Link]

I ended up with Dave Chinner instead of "XFS folks" or something like that exactly because Dave would also mentor new maintainers and hand over duties. So covers both the lack of review I see in the kernel at large, and the lack of ramping up new maintainers to keep up with the growth.

Reply to this comment

Vetter: Linux Kernel Maintainer Statistics
Posted Apr 29, 2018 4:36 UTC (Sun) by **zev** (subscriber, #88455) [Link]

Am I misreading something or is there something a bit screwy with the “kernel w/o GPU” graphs? Looking at the bars for v4.16 in Figure 3 for instance, I’d eyeball that at around 1100 reviewed out of 2400 total, or about 46%. The corresponding yellow bar in Figure 4, however, only appears to be around 37%. Likewise, from Figure 3 it would appear that the percentage of reviewed maintainer self-commits was slightly lower for v4.15 (~1100/2500) than v4.16, but the corresponding bars in Figure 4 appear to show v4.15 slightly higher than v4.16.

Reply to this comment