主　题:	Re: A way to reduce your workload: how do you think of it?	
发件人:	[redacted]	2018-7-30 23:14:32
收件人:	'[redacted]	
附　件:	signature.asc	

On 7/27/2018 3:11 AM, 谭鑫 wrote:

> [redacted]
>
> I am Xin Tan, a phd.student from peking university. I am doing a research about "how to reduce the burden of the linux kernel maintainers' work". I have some question that need your help.
>
> I know you are the maintainer of "INFINIBAND SUBSYSTEM". The subsystem of "INFINIBAND SUBSYSTEM" is a relatively active project. I observed that the number of the patches you signed off is very large, e.g, in the recent year, 1086 patches were signed off by you. I guess you must be very busy. In our study, we found several subsystems are using the new model to review the patches in order to reduce the maintainers' pressure. In particular, we studied i915 subsystem, and it is using the multiple-committer model and it seems run well so far. Pls. allow me give a brief introduction to this model. In traditional workflow, only the maintainers have the right to commit the patches to the repositories from developers, whereas multiple-committer model gives the commit right to the regular contributors (usually they are driver engineers who need to do core changes), so they can commit the patches to the same repository as the maintainer. We evaluated the effect of this model by the metrics we defined. We found it can significantly reduce the maintainers' workload, latency and overwork, which makes review work more balanced to keep maintainers from burning out. Meanwhile, because it introduces a stricter review mechanism (e.g., the patches submitted by committers need to be reviewed by at least another one developers), the quality of patches is also guaranteed.
>
> We also analyze what types of the subsystems are suitable to use this model. The subsystem of "INFINIBAND SUBSYSTEM" is one of the subsystems we think is suitable to use this model. We have the following reasons:
>
> 1) its maintainer is very busy (rank 3/359 in the recent year)

I have a co-maintainer that also commits to this tree. He is Cc:ed on this email.

> 2) the code ownership is relatively low (rank 71/367 in the subsystems we analyzed) which means that multiple developers modify the same file at the same time. Using multiple-committer model could avoid the conflict because all the patches are submitted to the same tree. It also may suggest that developers trust each other.

No, this would not be the case. This subsystem has lots of developers that maintain a fairly contentious relationship. Even though they modify the same file, I would not say that they trust each other or that they would be willing to allow their fellow developers to have write access to the tree. One of the things Jason and I do is act as an arbiter whenever they have unresolved conflicts. Being gatekeepers of the code going into the tree makes that arbitration easier.

> 3) developers are sharing affiliations (Mellanox and Intel contributes 57% of patches and 44% of developers), which suggests that these developers are more trustworthy compared to subsystems with disperse developers. Therefore, it is possible to find the right developers as committers.

Having a large number of developers working on a subsystem does not mean those developers are trustworthy, it merely means that the company has interests in this area and is funding work on it. You can't draw the conclusion you did from the data at hand. Intel and Mellanox are locked in competition in the marketplace and each has a different model that their hardware follows. Changes Intel needs for their hardware might be detrimental to Mellanox hardware (I'm saying there is malice or intent here, merely that one person's changes to the core code might negatively impact the other). On a regular basis Jason and I have to check for these sorts of things. So it's not even that the developers aren't trustworthy, it's that they are both off doing their own thing, their hardware is drastically different, and their core changes might or might not be good for the rest of the vendors out there, so we have to stand as gatekeepers for this issue.

> How do you think of this model? We would like to know your opinion.
> Thank you very much!

I don't think it's a good idea in our subsystem, but I'll allow Jason to speak his mind on the topic.