Can reviews become less of a bottleneck?

Kristof Beyls

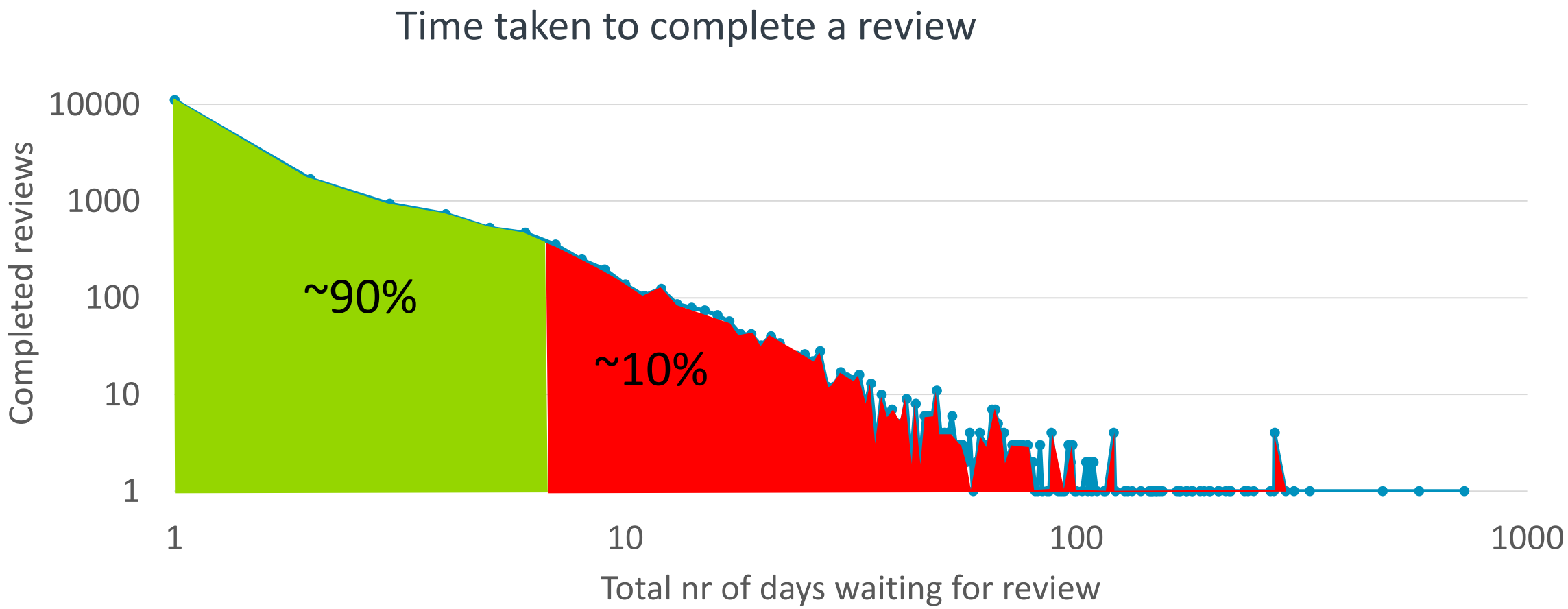
Lightning talk – EuroLLVM 2018

Some characteristics of reviews on reviews.llvm.org

arm

1. How long does a contributor have to wait for review?

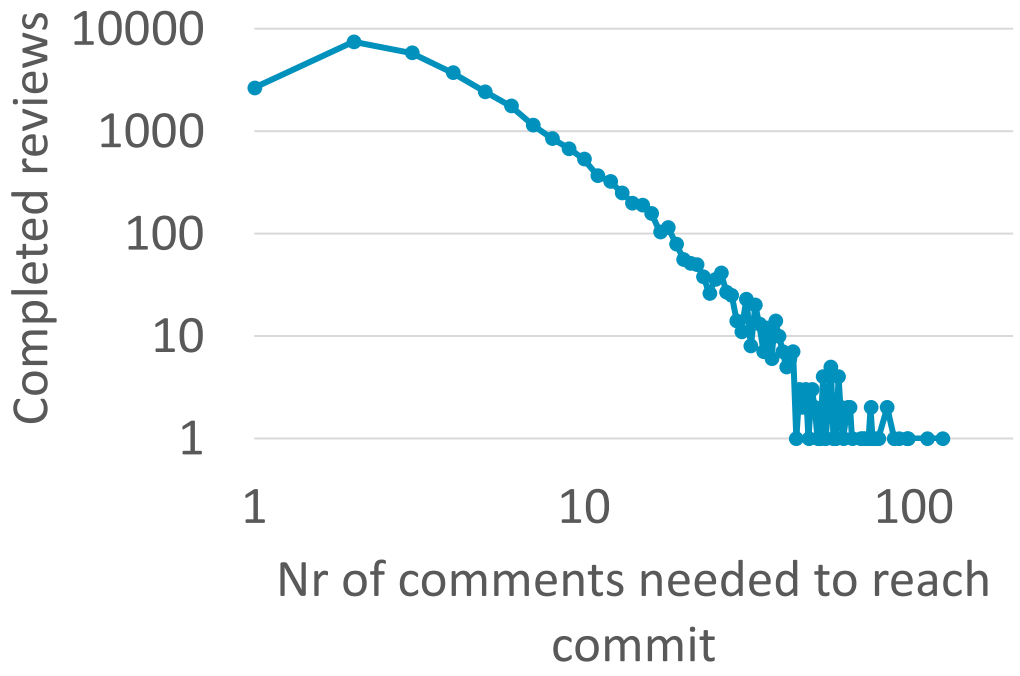
In how far were reviews the bottleneck (mid 2015 – early 2018)?



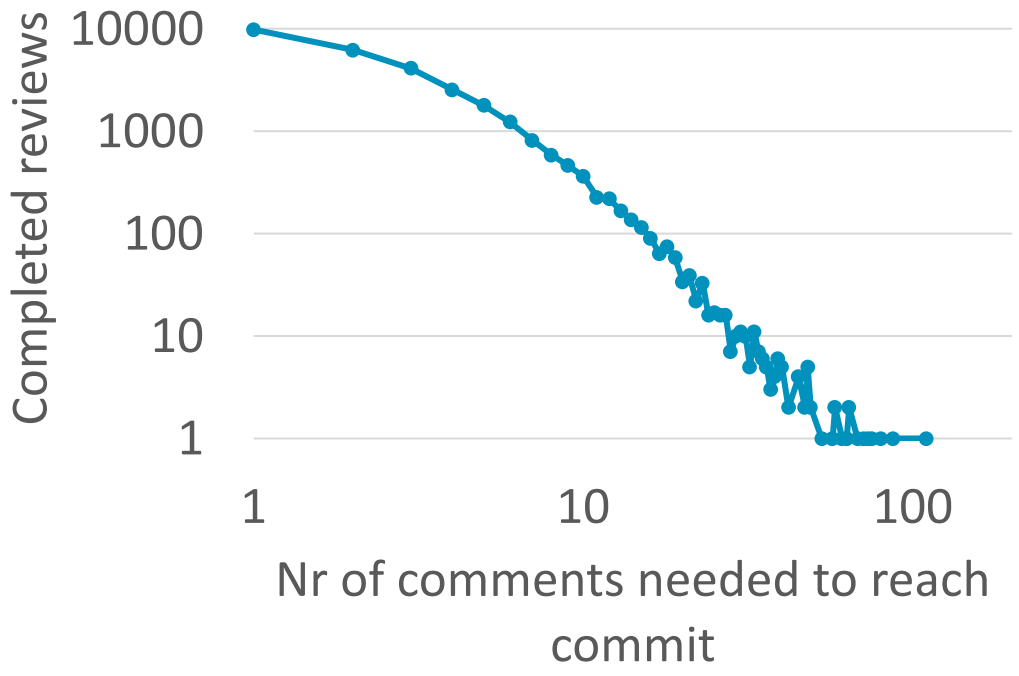
arm

2. How many feedback rounds are needed before committing?

On average, a review needs 3.8 comments



Of which 2.5 comments from outside your organization



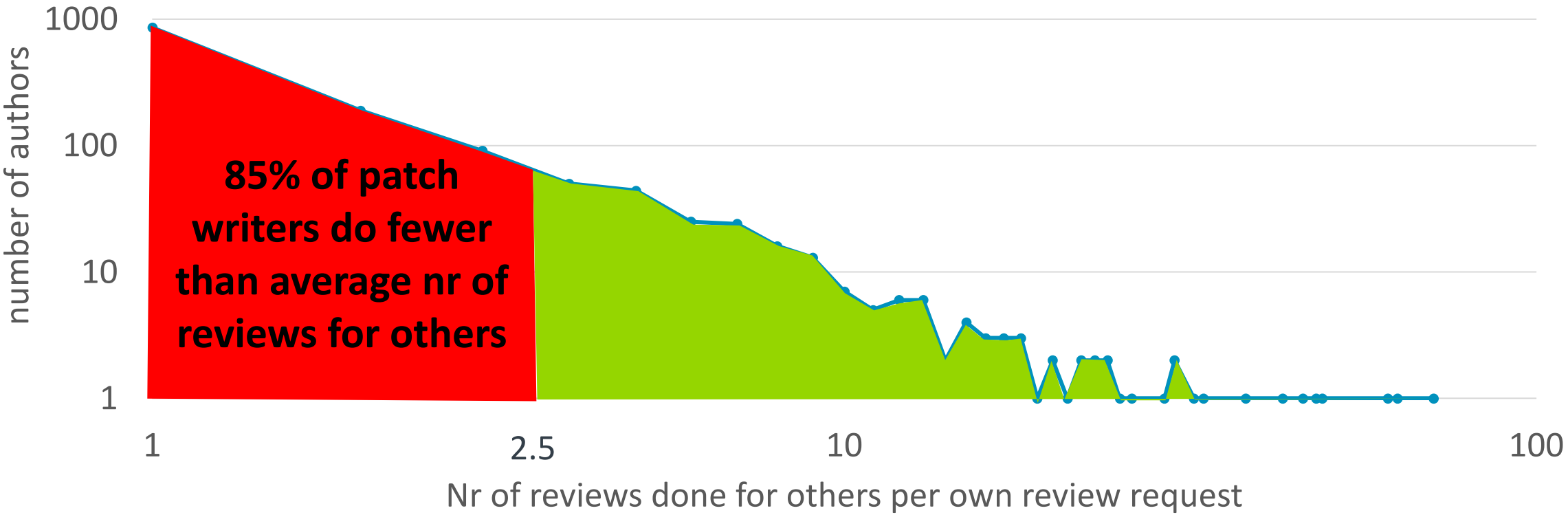
arm

For every review you request yourself,

you should aim to give back at least 2.5 insightful comments for others, to not be in “review debt”.

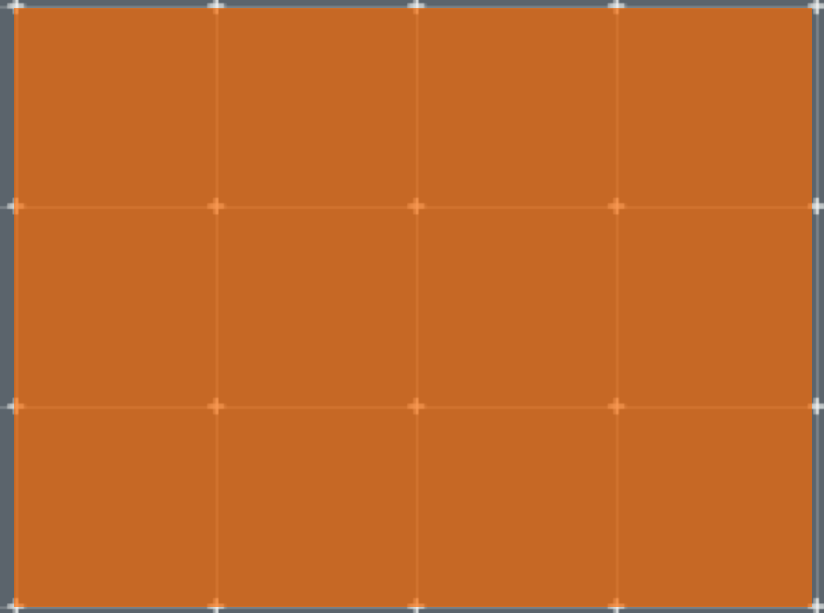
arm

3. How many reviews for others do people do?



arm

Making reviews less of a bottleneck….
By getting more people to do reviews?



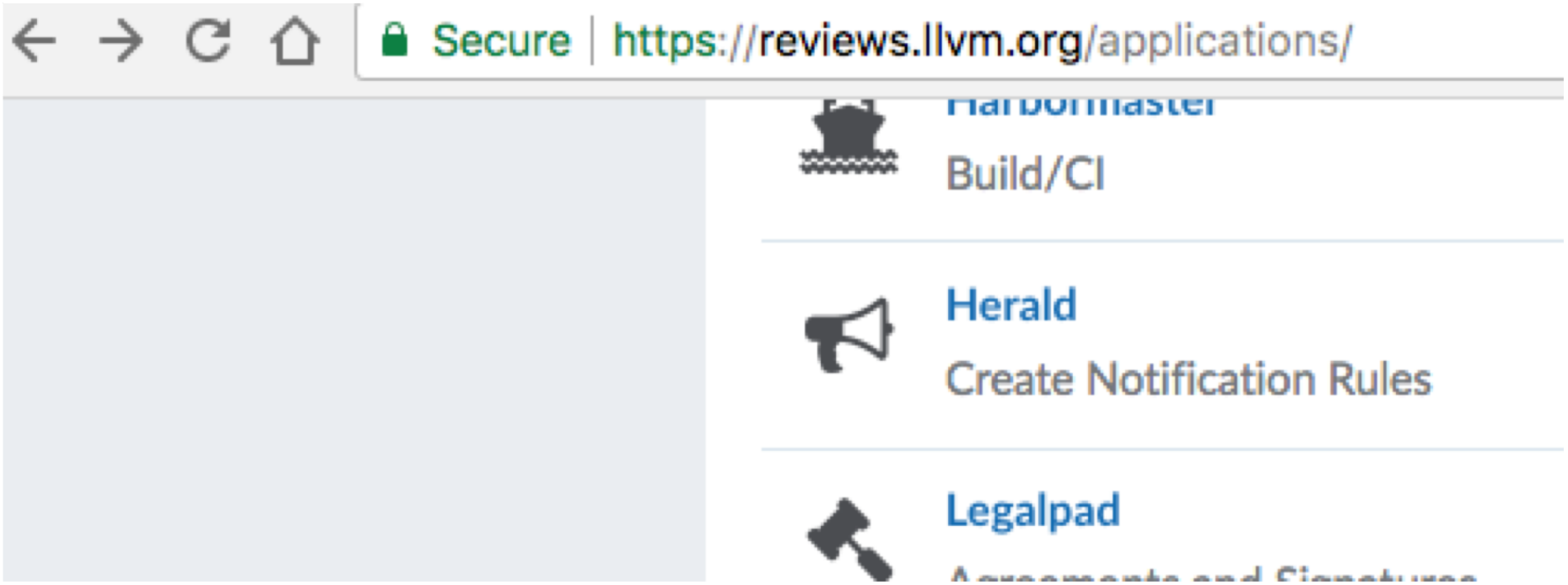
arm

What stops people from doing reviews for others?

1. Maybe lack of understanding of how many reviews are needed from you to keep the community going?
   - Guideline: try to do at least 2.5 reviews for others for every review you request yourself.
   - Some can only contribute to downstream/private forks of LLVM.
     But you could still perform upstream reviews for others in your area of expertise?
2. Over 250 reviews every day where something changes.
   How can I find the few that I can contribute to in a reasonable time?
   - 2 suggestions on following slides

1. Use Herald (Phabricator application)

… to get added as a subscriber onto reviews with keywords that you have an interest in.



arm

2. Add custom scripting to heuristically find patches

For example, patches that touch on lines of code you touched before:

```
SUMMARY FOR kristof.beyls@arm.com (found 4 reviews):
[9.52%/4.76%] https://reviews.llvm.org/D42377 '[CodeGen] Use MIR syntax for MachineMemOperand printing'
by Francis Visoiu Mistrih
[0.00%/40.00%] https://reviews.llvm.org/D43374 '[ARM]Decoding MSR with unpredictable destination register
causes an assert' by Simi Pallipurath
[0.00%/33.33%] https://reviews.llvm.org/D44128 '[GISel]: Add helpers for easy building G_FCONSTANT along
with matchers' by Aditya Nandakumar
[0.00%/8.33%] https://reviews.llvm.org/D44043 '[DAGCombine] Remove AND in SETCC if we can prove they are
unneeded' by Dave Green
```

Conclusion

- When asking for review:
  - Add the right people to the review. See https://llvm.org/docs/Phabricator.html for some advice.
  - Alex Bradbury's weekly contains a "patches needing review" section. Make use of it if your patch gets stuck.
- Do your own fair share of reviewing:
  - Aim to at least do 2.5 reviews for others for every patch you contribute.
  - When you don't contribute code upstream, still consider doing code reviews upstream.
- As a potential reviewer, use tools to find reviews-in-progress you can contribute to.
  - The Herald application in Phabricator is an easy way to do so
  - Only scratched the surface on what further automation could be done.
    Should we add more automation to help highlight reviews you could help with?

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद