Bridging the gap in back-porting Linux kernel patches.

Akshith Gunasekaran
School of EECS, Oregon State University
Corvallis, USA
gunaseka@oregonstate.edu

***Abstract*—**

***Index Terms*—**

I. Introduction

A. Problem statement

Back-porting patches from the mainline kernel to the downstream kernels is a high-effort, high-skilled task. This work will explore the challenges to automating this process which is critical to the Linux kernel ecosystem.

B. Motivation

There are n different distributions of Linux specialized for different use cases. For example Arch and Fedora prioritize new features, which Debian and RedHat prioritize reliability and stability. But all developments (new features, bug fixes, refactoring) happen on the mainline kernel. So when new feature/improvements/fixes appear distro maintainers need to add the *relevant* patches from the mainline kernel to their respective kernel.

The rate at which patches land in the mainline far exceeds the rate at which maintainers can port it to their kernel. The task boils down to the following subtasks.

- What patches should I port?
- Is the patch relevant to my kernel?

Existing literature [1] shows that the first subtask has been addressed by machine learning tools like AUTOSEL [2] and other Distribution specific tools. But the second subtask is still highly manual. It requires the developer to understand the patch and see if the patch is applicable to a specific kernel version they are maintaining.

C. Relation with software engineering research

There has been multiple measurement studies [1], [3], [4] in SE research that have quantified the patch porting practices of the Linux kernel ecosystem and other software projects. But there is a large gap yet to be addressed.

D. Key Insight or Idea

The key insight comes from the fact that this gap is largely due to lack of understanding of the context of the patch. Especially when the file names do not match hard to locate the context of the patch in earlier versions of the kernel.

Given that LLMs have shown to be effective in understanding the context of the code and have been effective in supporting developers in various tasks like code completion, code summarization and code search we believe that LLMs can be a useful tool in understanding the context of the patch and thus automating the process of back-porting patches.

I plan on building on top of existing tools that support majority of the pipeline which should allow me to focus on the core problem of understanding the context of the patch.

E. Assumptions

F. Research questions

[[What research question(s) will you answer?]]

RQ1. Can LLMs be used to understand the context of the patch?

RQ2. Can LLMs be used to automate patch transplantation?

G. Evaluation Dataset

- Dataset containing patches from the mainline kernel and the corresponding downstream kernel patches.
- Dataset of CVEs and the patches that fix them.

The dataset already exists in an existing tool called FixMorph [5].

H. Evaluation metrics

[[How will you know that you have solved the problem successfully? In other words, how will you evaluate your solution?]]

I plan to evaluate the success the model has in understanding the context of the patch by comparing the context that the model thinks are relevant to the actual context in the dataset. I plan to use the following metrics to evaluate the model:

- Precision
- Recall
- F1 score

II. Background and Motivation

III. Related Work

IV. Approach

V. Evaluation

A. Dataset

B. Metrics

C. Experiment Procedure

D. Results

VI. Discussion and Threats to Validity

- statistical significance tests

VII. Contributions

References

[1] X. Li, Z. Zhang, Z. Qian, T. Jaeger, and C. Song, "An investigation of patch porting practices of the linux kernel ecosystem," *CoRR*, vol. abs/2402.05212, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.05212

[2] J. Edge, "Automating stable-kernel creation," *LWN.net*, 2016. [Online]. Available: https://lwn.net/Articles/701304/

[3] D. Chakroborti, K. A. Schneider, and C. K. Roy, "Backports: change types, challenges and strategies," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, ser. ICPC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 636–647. [Online]. Available: https://doi.org/10.1145/3524610.3527920

[4] R. Shariffdeen, X. Gao, G. J. Duck, S. H. Tan, J. Lawall, and A. Roychoudhury, "Automated patch backporting in linux (experience paper)," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 633–645. [Online]. Available: https://doi.org/10.1145/3460319.3464821

[5] "Fixmorph - program transformation, patch backporting, code transplantation," https://fixmorph.github.io/, 2021.