SYSYPHUZZ: the Pressure of More Coverage

Zezhong Ren[‡§], Han Zheng[§], Zhiyao Feng[§], Qinying Wang[§], Marcel Busch[§]
Yuqing Zhang[‡], Chao Zhang[†], Mathias Payer[§]

[‡]UCAS [§]EPFL [†]Tsinghua University

EPFL

hexhive

SYSYPHUZZ: the Pressure of More Coverage

Motivation: why coverage alone is not enough

Key observation: extreme imbalance in execution frequency

SYSYPHUZZ design: Boost Delegator & Context-Preserving Mutation

Evaluation: low-frequency coverage and bug discovery improvements

Takeaways and **future directions**

Motivation: Coverage Alone is Insufficient

Kernel fuzzers optimize for maximizing **unique** coverage.

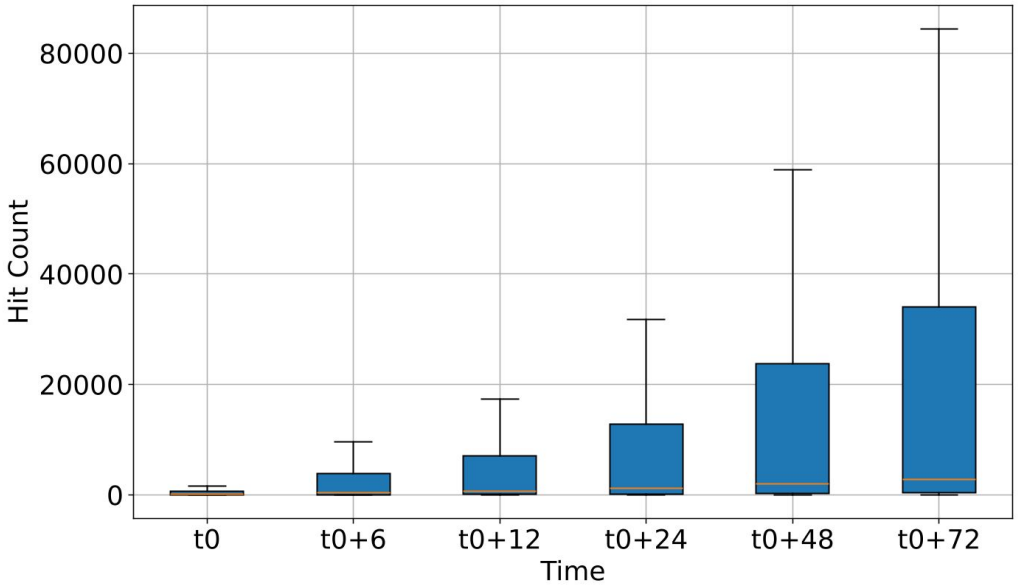
Coverage ≠ thorough exploration: **hot paths** can dominate execution.

Existing corpora (e.g., Syzbot's) make finding "**new coverage**" difficult.

A large fraction of kernel code is reached **rarely,** even after long campaigns.

Hypothesis: under-explored, low-frequency regions **hide** overlooked bugs.

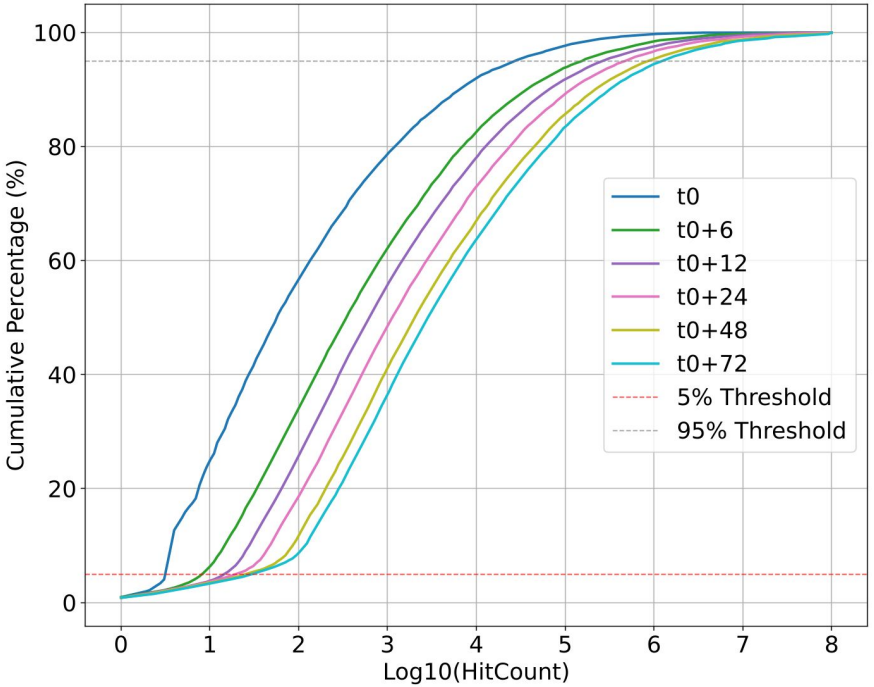
Observation 1: Execution Frequency is Highly Imbalanced.



Hit-count distribution over time (box is the interquartile range; orange line is median).

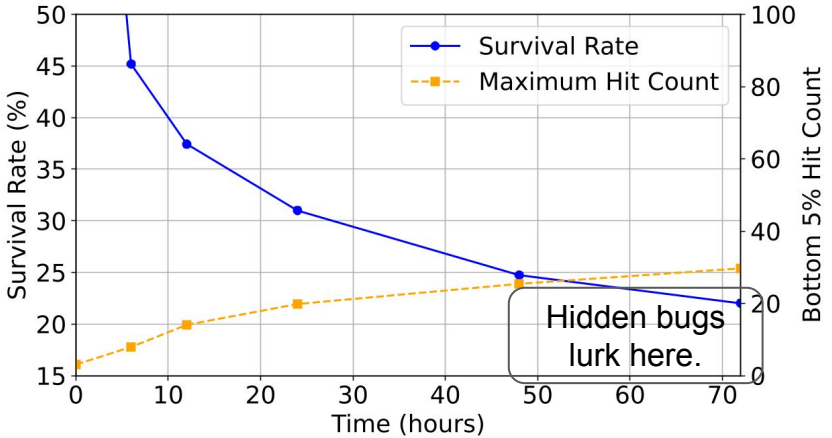
Coverage frequency is highly skewed: 50% of covered BBs have much lower hit counts than the rest. **5%** of BBs are hit fewer than **30** times in the end.

Observation 2: Some BBs Remain Persistently Overlooked



Left: CDF of BB hit counts (log scale), Syzkaller.

Bottom: Survival rate of t0 bottom-5% BBs over time + their maximum hit count.



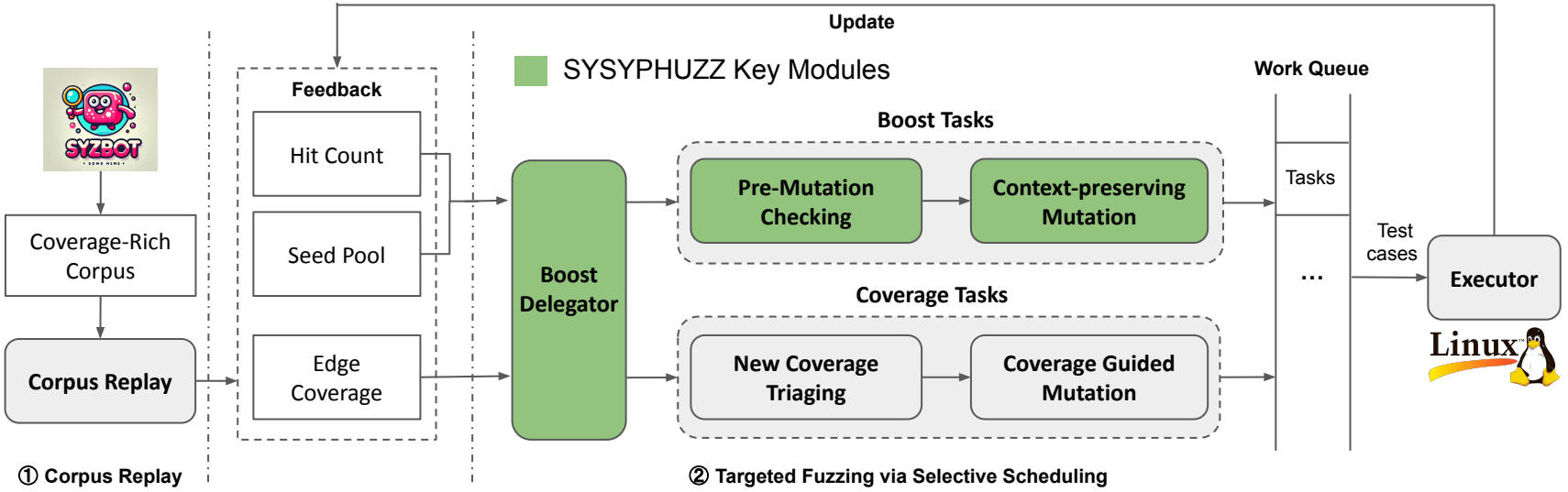
SYSYPHUZZ Design

We aim to help the fuzzer identify and explore **low-frequency regions** more thoroughly **without** sacrificing overall coverage growth.

Challenge 1: **Resource Constraints**
Boosting low-frequency regions adds extra **tasks** (fuzzing jobs) guided by hit-frequency signals, which can **overwhelm resources** or **starve coverage-driven exploration**.

Challenge 2: **Context-destroying Mutations**
Low-frequency regions often depend on **precise syscall sequences**. Random mutations frequently **break** this context, making the targets **unreachable**.

SYSYPHUZZ: Overview



Stage 1. Corpus Replay

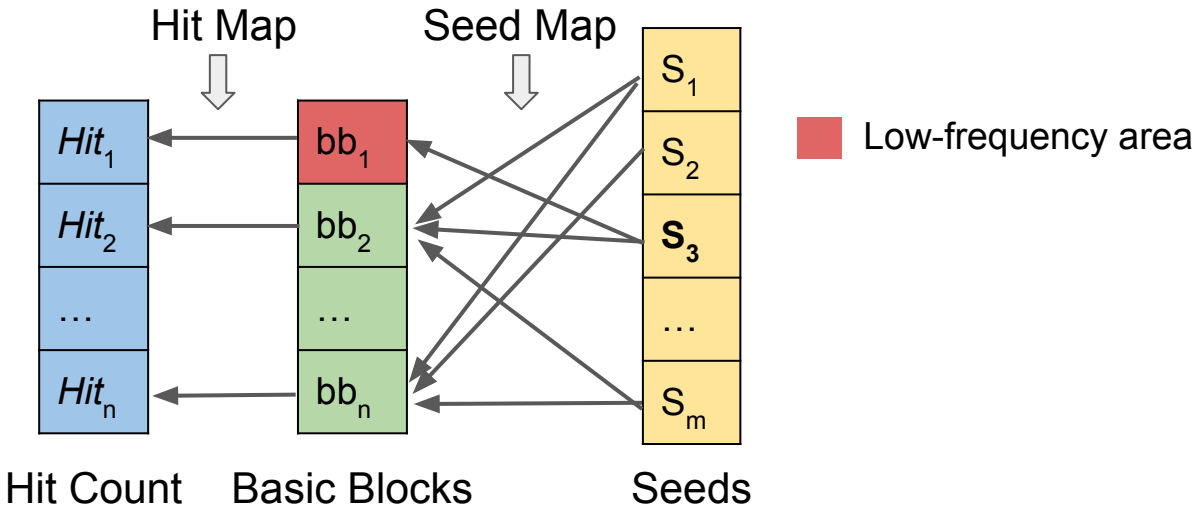
Replay the syzbot corpus.

Collect BB hit counts.

Stage 2. Low-Frequency-Guided Exploration

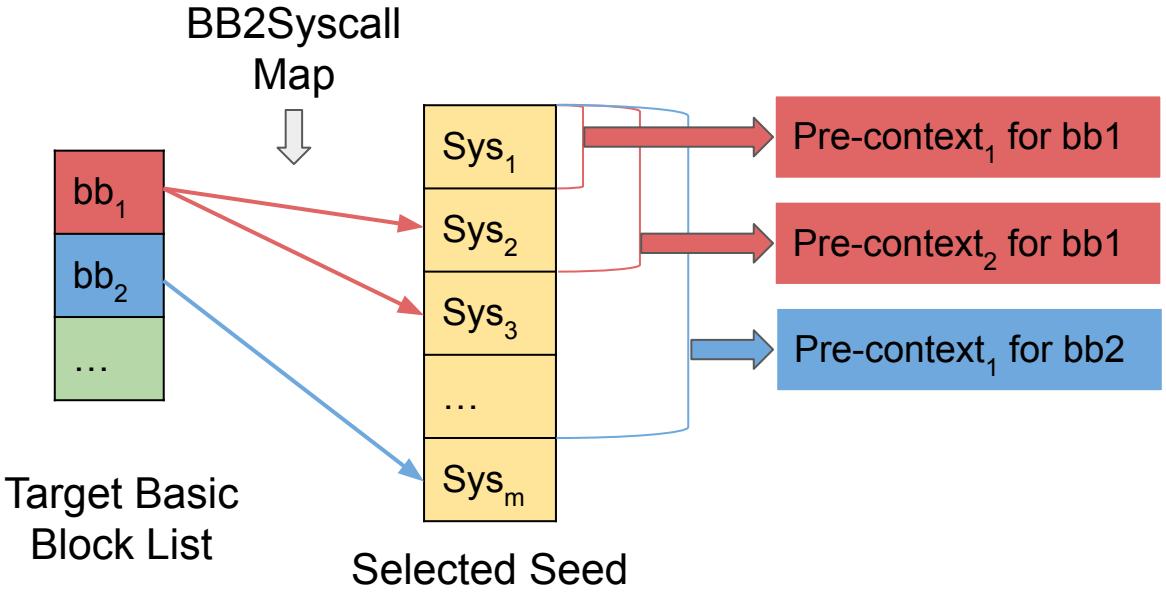
The Boost Delegator selects target BBs, generates and schedules boost tasks, and applies **context-preserving mutations**.

SYSYPHUZZ: Seed Selection



After each boost round (i.e., all boost tasks in the queue finish), the **boost delegator** selects target regions (e.g., *bb1*) and uses the **seed map** to select initial seeds (e.g., *S3*) for the next round.

SYSYPHUZZ: Context-Preserving Mutation



The **pre-mutation checking** identifies the **syscall(s)** that hit the target BB (e.g., Sys2, Sys3 for bb1). The preceding syscall sequence forms the **pre-context**—the required dependency to reach the target (e.g., Sys1; Sys1→Sys2).

Evaluation: Setup and Key RQs

Targets: Linux kernel 6.12-rc6.

Baselines: Syzkaller and SyzGPT.

Campaign length: 72 hours.

Corpus: Established via Syzbot.

Metrics:

– Basic block hit-count

– Survival rate

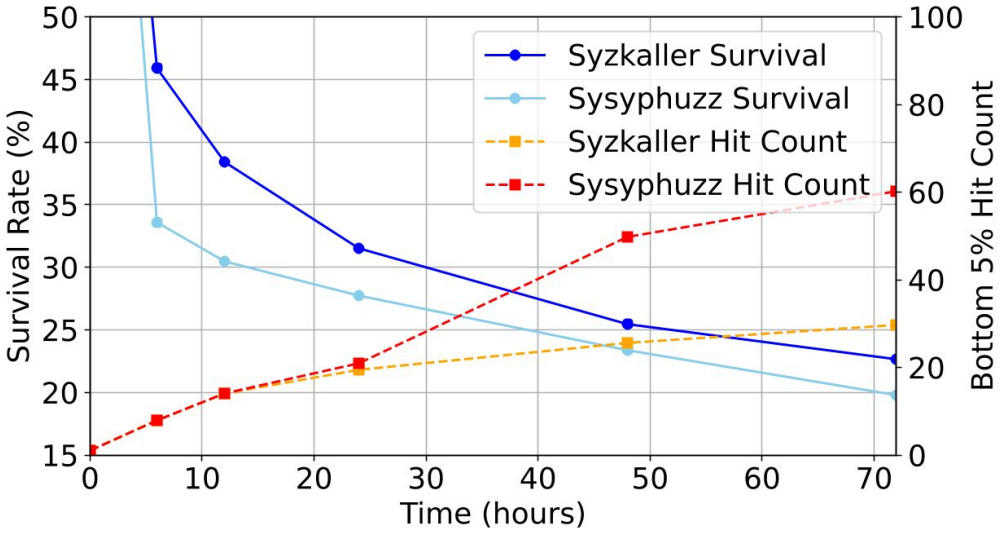
– Coverage and unique bugs

RQ1: Can we increase the hit counts of persistently overlooked low-frequency regions?

RQ2: Can we outperform state-of-the-art kernel fuzzers?

RQ3: What is the relationship between bug discovery and targeted low-frequency regions?

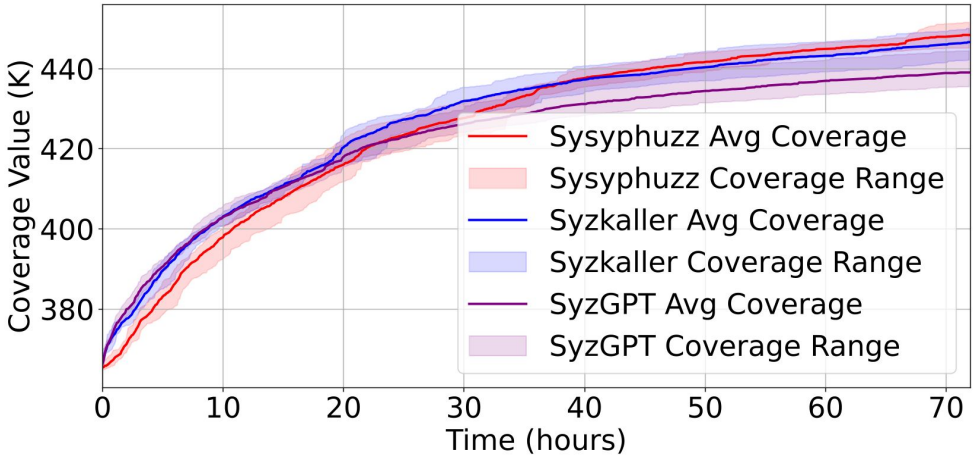
RQ1: Hit Counts of Low-frequency Kernel Code

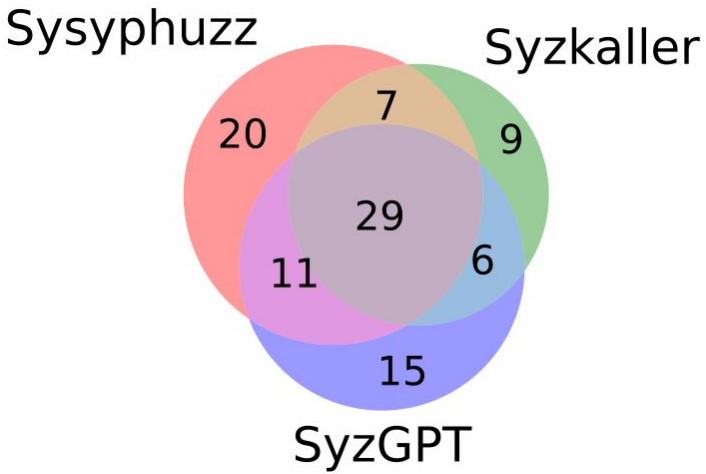


Survival rate of the t0 bottom-5% BBs—i.e., the fraction that remain in the bottom 5% at later times—and their maximum hit counts.

Compared to **Syzkaller**, after 72 hours **SYSYPHUZZ** lowers the survival rate (**22.0%** → **19.3%**) and raises the bottom-5% max hit count (**<30** → **~60**).

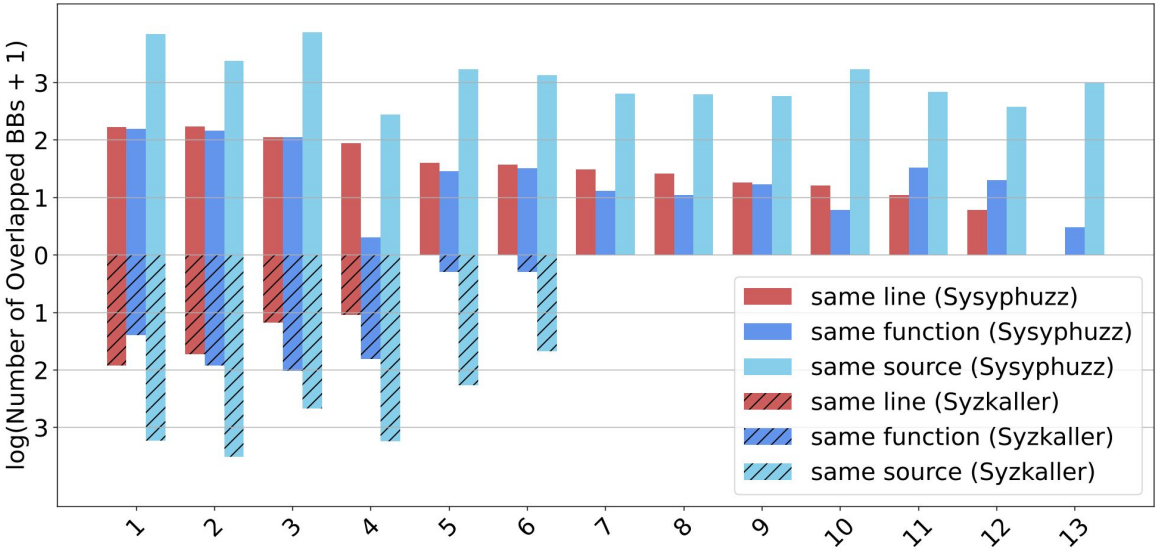
RQ2: Compare to SOTA





SYSYPHUZZ gains **+8k edges** over Syzkaller (p=0.07) and **+20k** over SyzGPT (p=0.0025). It finds **67 bugs—+31%** vs. Syzkaller (p=0.03) and **+9.8%** vs. SyzGPT (p=0.08).

RQ3: Overlap between Bugs and the Low-freq Areas



Left: Overlap between SYSYPHUZZ-identified low-frequency BBs and sanitizer bugs discovered exclusively by SYSYPHUZZ vs. Syzkaller.

A bug is **low-frequency–related** if it covers ≥1 low-frequency BB (**same line category**): SYSYPHUZZ is **~6×** more likely to find such bugs. On average, SYSYPHUZZ bug stack traces contain **~1,000** more low-frequency BBs than Syzkaller's.

Future Directions

Adaptive threshold: **Tune** the bottom-5% cutoff to balance target size and boosting effectiveness; aim for auto-tuning across kernels/workloads.

Corpus sensitivity: Study different initial corpora (syzbot vs. empty/smaller) to **quantify** warm-start effects and long-run convergence.

Hard-to-reach targets: DenyList reduces wasted effort, but schedule/mutation alone may not re-enter stalled regions quickly; selectively use higher-overhead methods (e.g., LLM agent) to **restore reachability**.

SYSYPHUZZ Conclusion

Frequency imbalance is a blind spot of coverage-centric kernel fuzzing.

- SYSYPHUZZ introduces **Boost Delegator** and **Context-Preserving Mutation** to (more) thoroughly explore low-frequency regions.
- SYSYPHUZZ delivers **~2×** low-frequency coverage, fewer persistently overlooked regions, and more unique bugs.

Execution frequency is complementary to **coverage** for systems fuzzing.

Source code: https://github.com/HexHive/Sysyphuzz

Contact: renzezhongucas@gmail.com

EPFL

hexhive