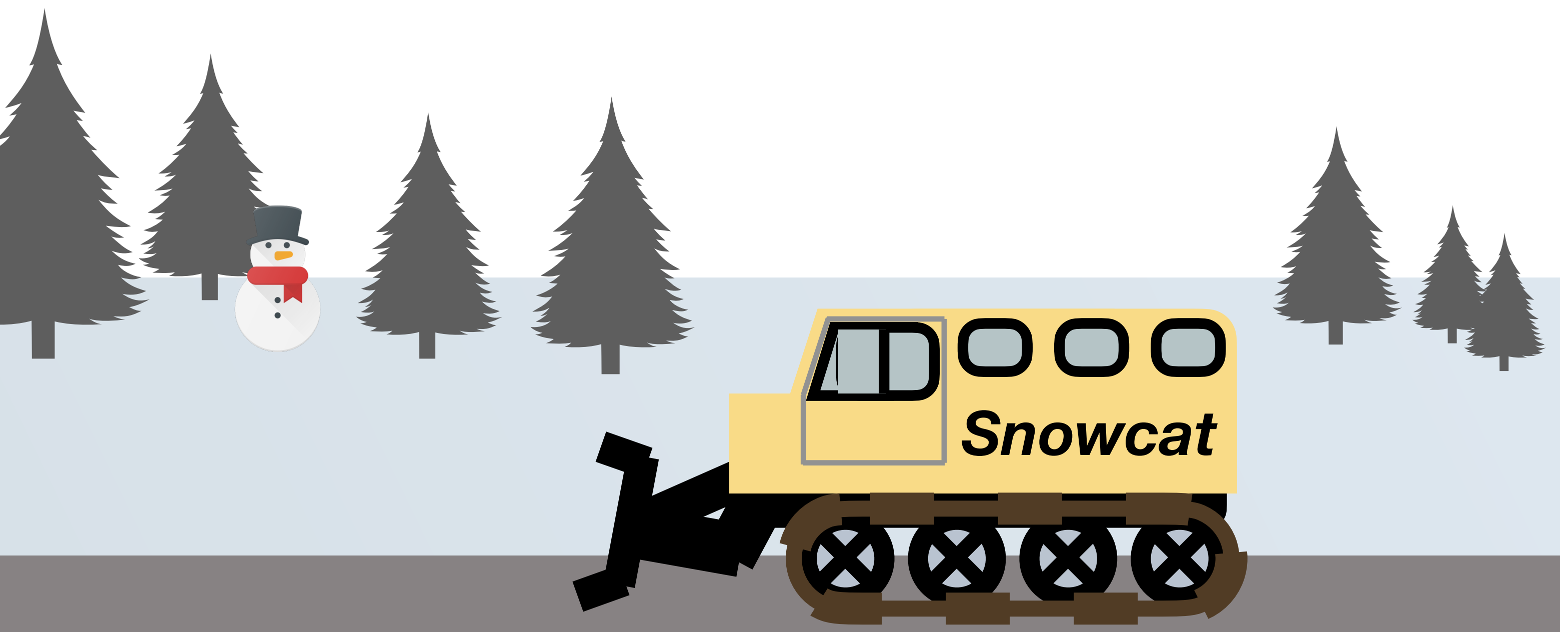
Snowcat: Efficient Kernel Concurrency Testing using a Learned Coverage Predictor

Sishuai Gong[1], Dinglan Peng[1], Deniz Altınbüken[2], Pedro Fonseca[1], Petros Maniatis[2]

[1]Purdue University [2]Google DeepMind



PURDUE UNIVERSITY

RELIABLE & SECURE SYSTEMS

Google DeepMind

Finding kernel concurrency bugs is important

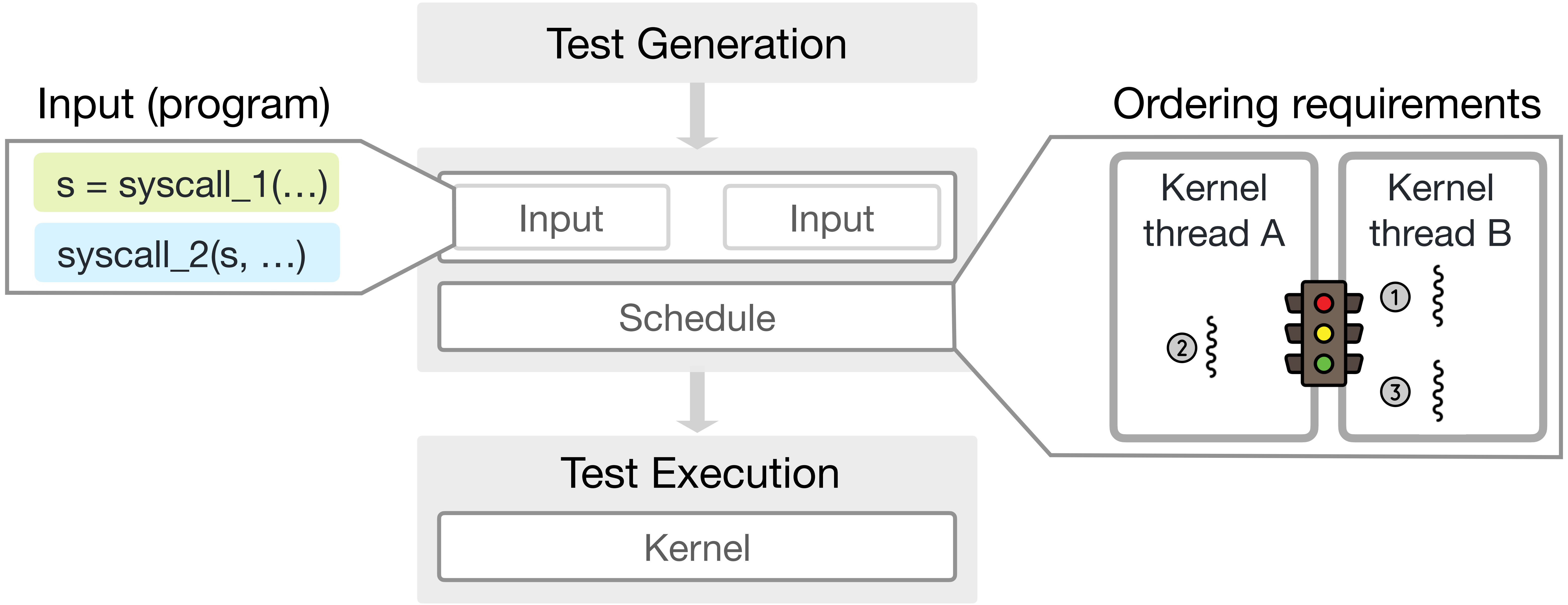
- Bugs that depend on the instruction schedules.
- Such bugs have serious impact.

The Hacker News

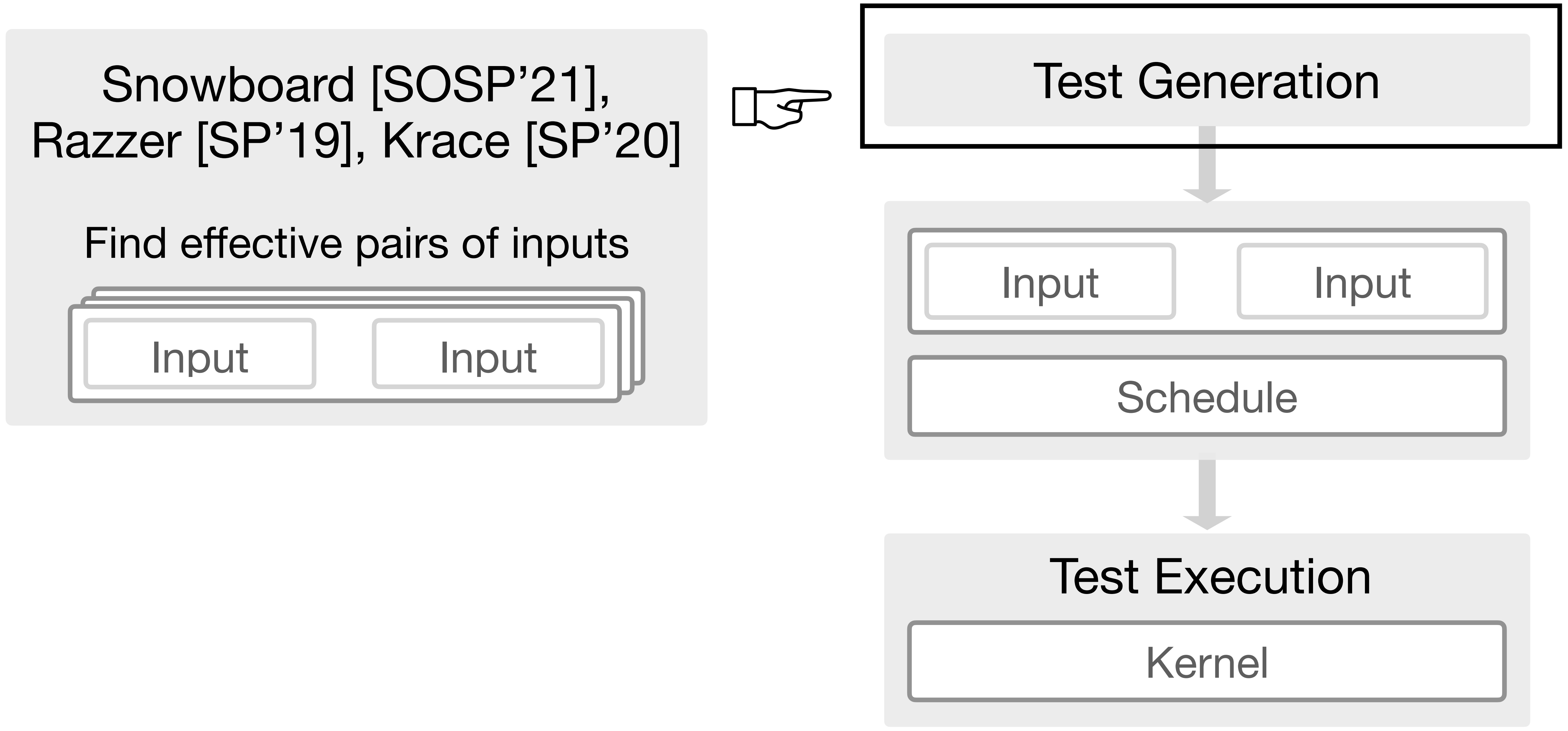
Researchers Uncover New Linux Kernel 'StackRot' Privilege Escalation Vulnerability

Root cause: a kernel concurrency bug

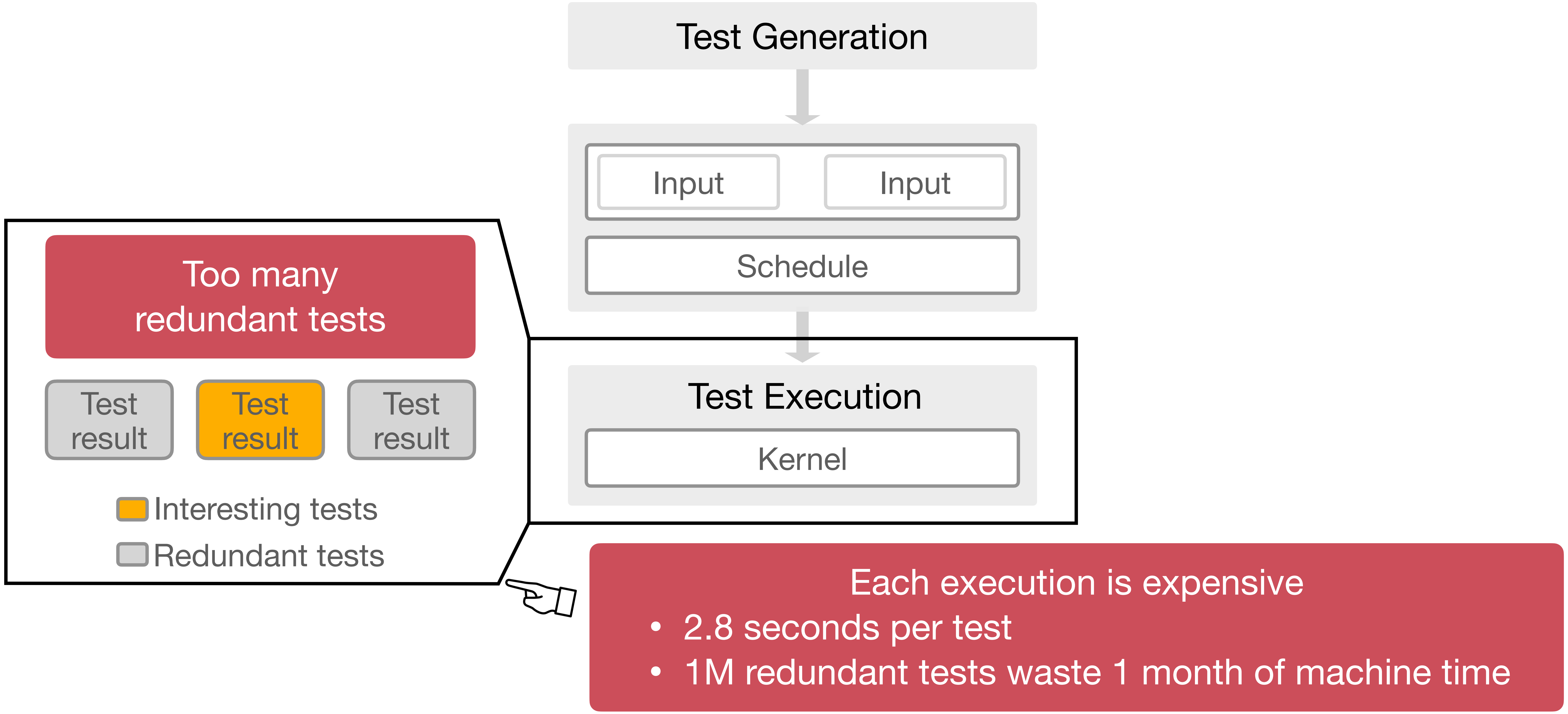
Find kernel concurrency bugs through testing



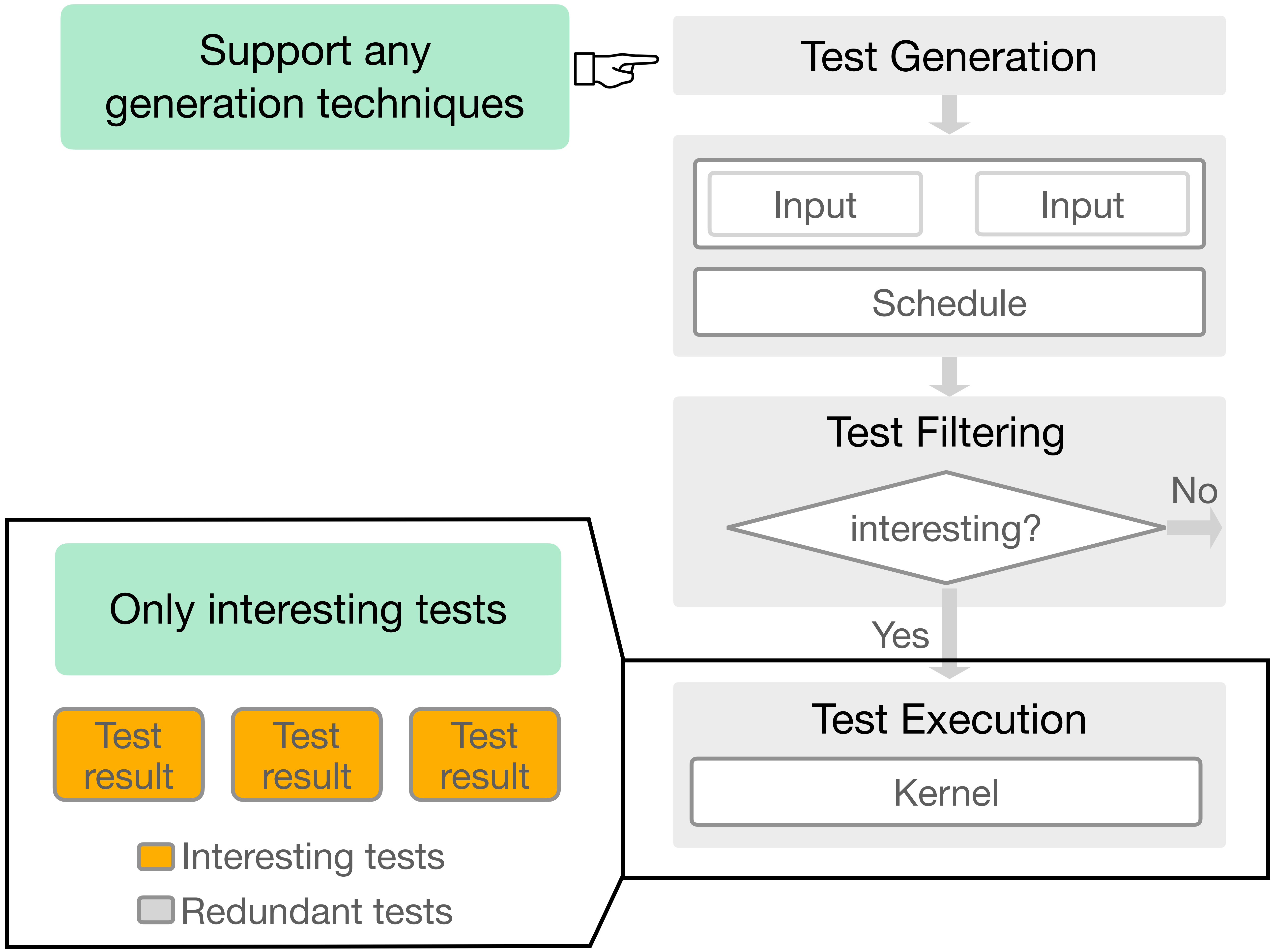
Prior work focuses on optimizing test generation



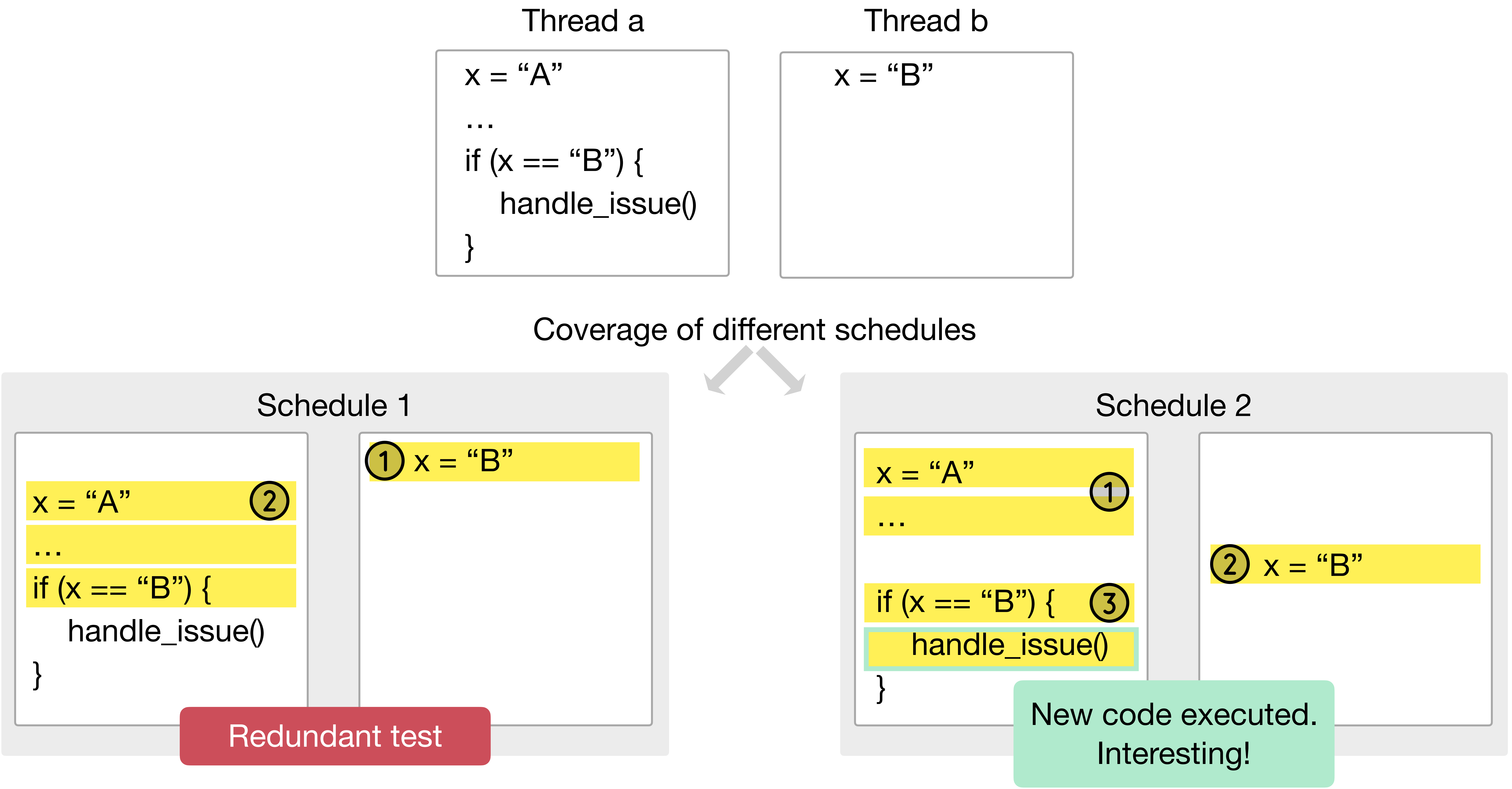
Redundant tests reduce the testing efficiency



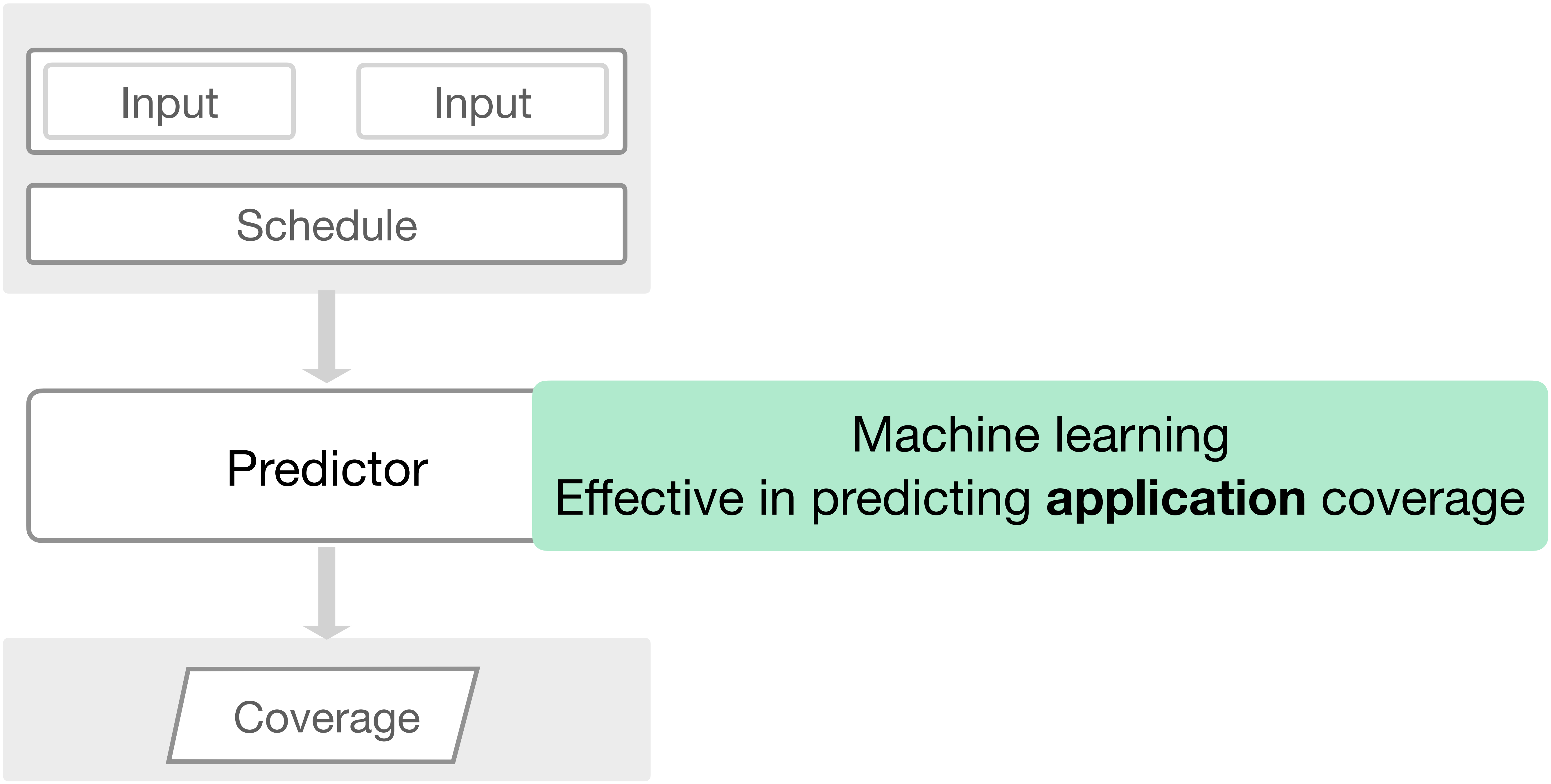
Identify and only execute interesting tests



What are interesting tests?



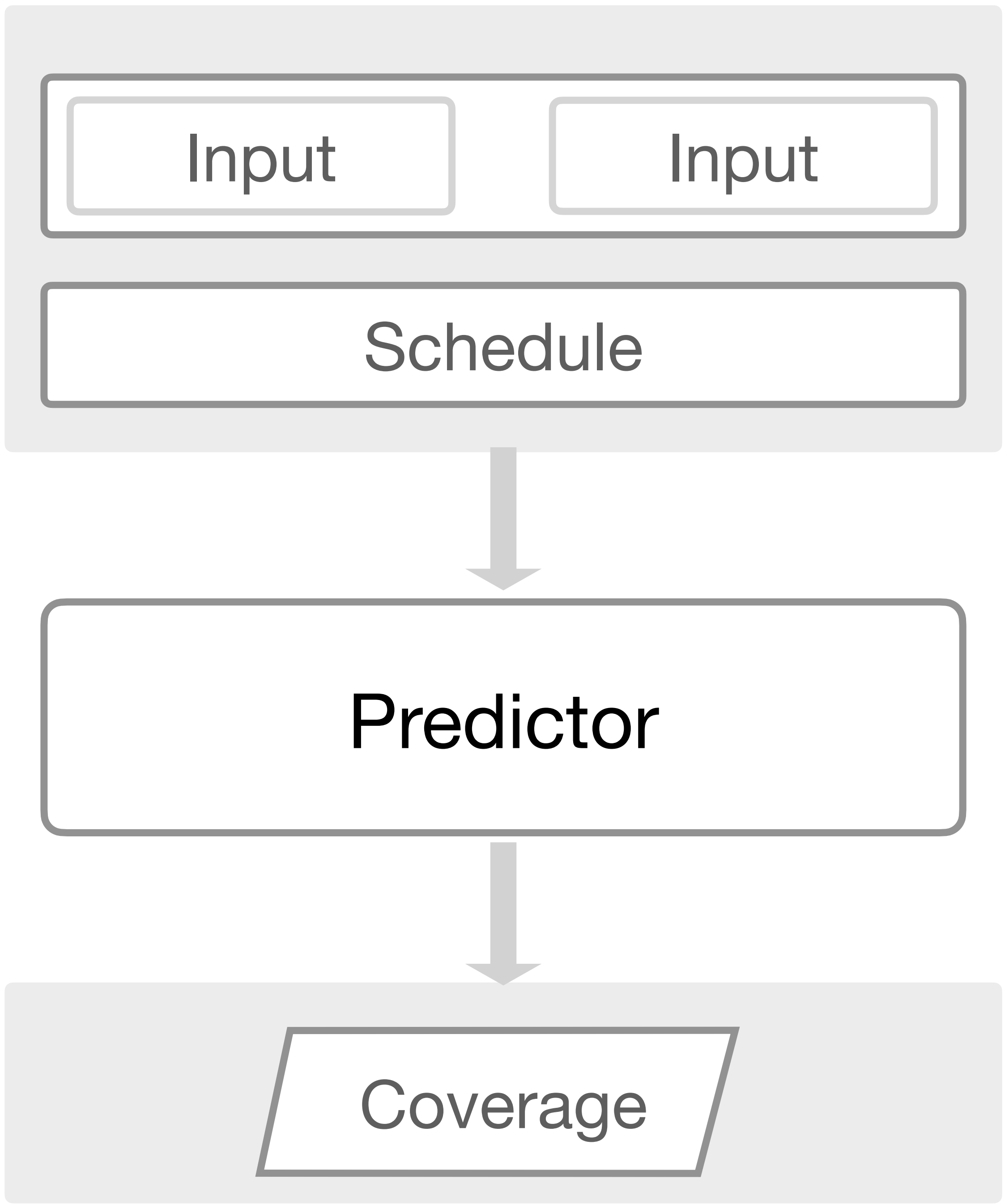
Snowcat predicts the kernel coverage for concurrency tests



Challenges in using ML to build the predictor

1. How to encode a concurrency test to the model?

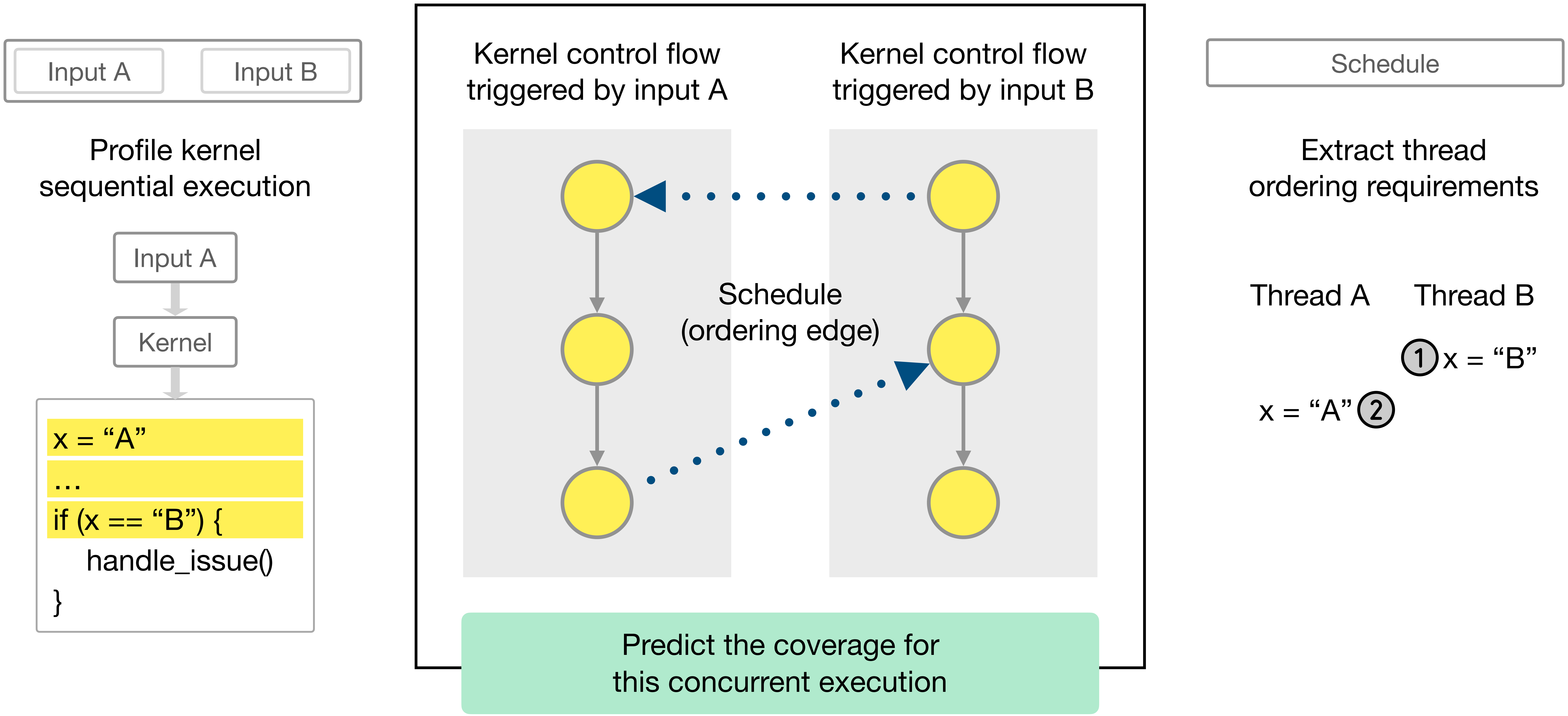
- Inputs are userspace programs
- Schedule is kernel threads ordering



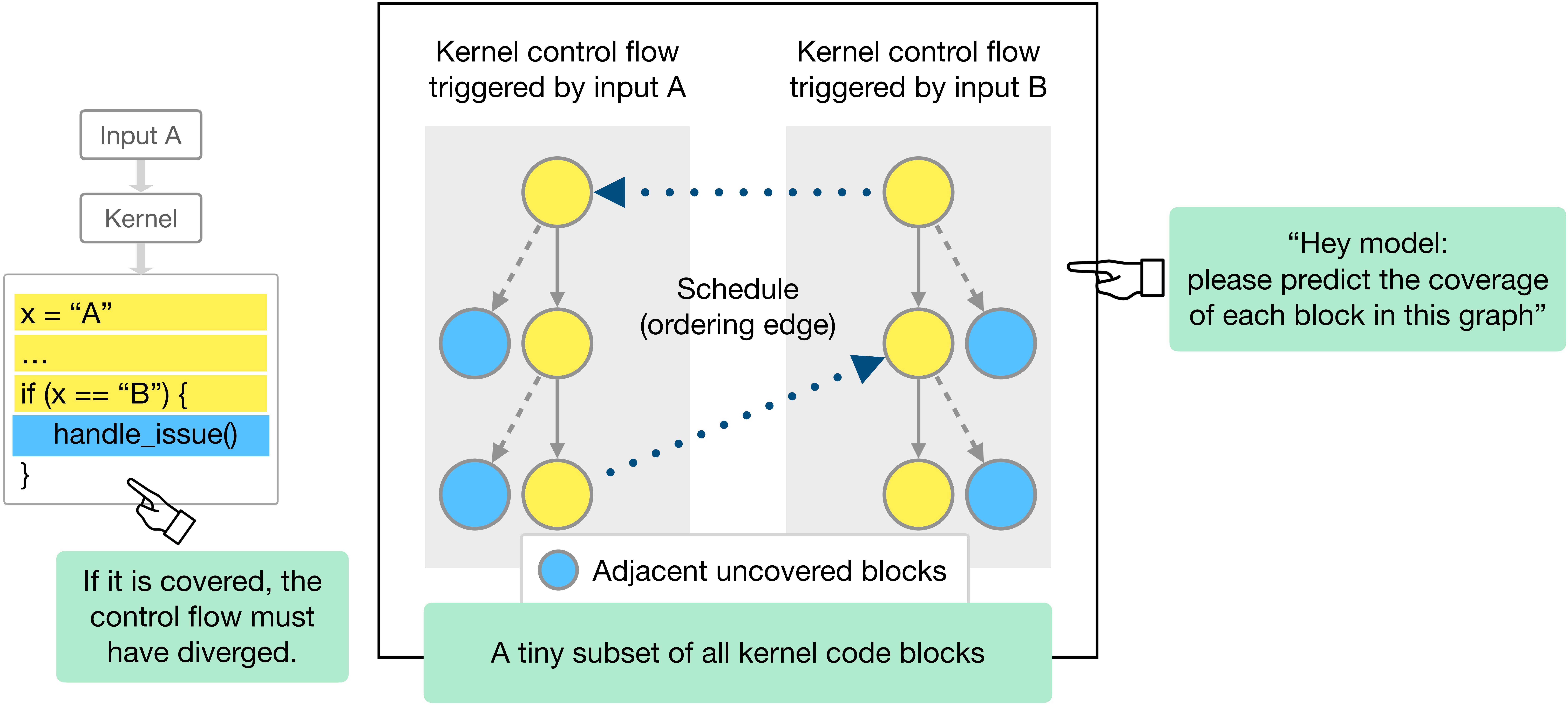
2. How to predict much faster than execute?

- Predicting a full kernel CFG takes ~3s.
- A concurrent execution takes ~2s.

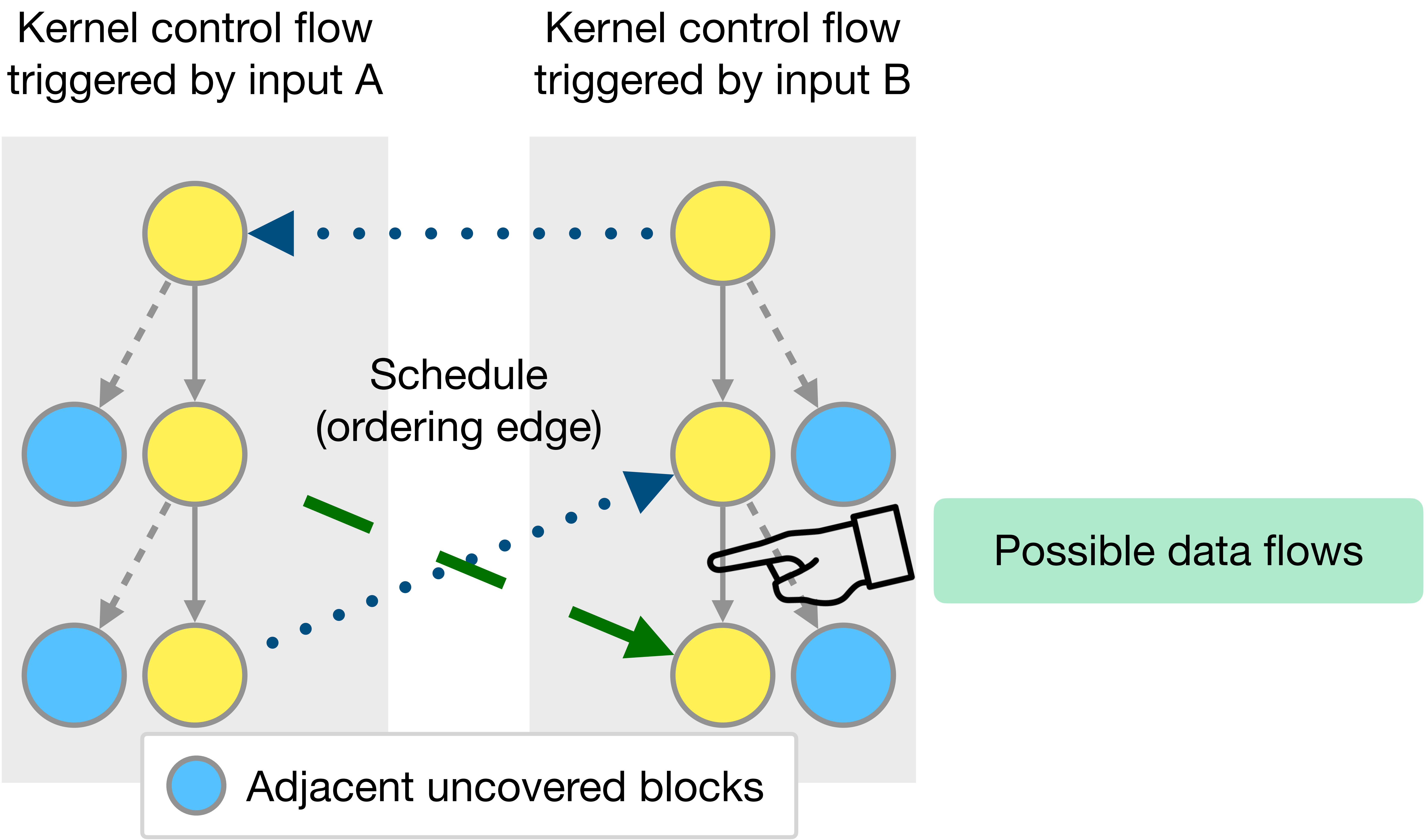
1. Represent the schedule on two sequential control flows



2. Predict faster by only considering adjacent uncovered blocks



Snowcat encodes even more information in the graph



Implementation of Snowcat

1. Build the predictor using ML

Training dataset

- Data

  1.3M tests and their coverage
- Kernel version

  Linux kernel 5.12

Model architecture

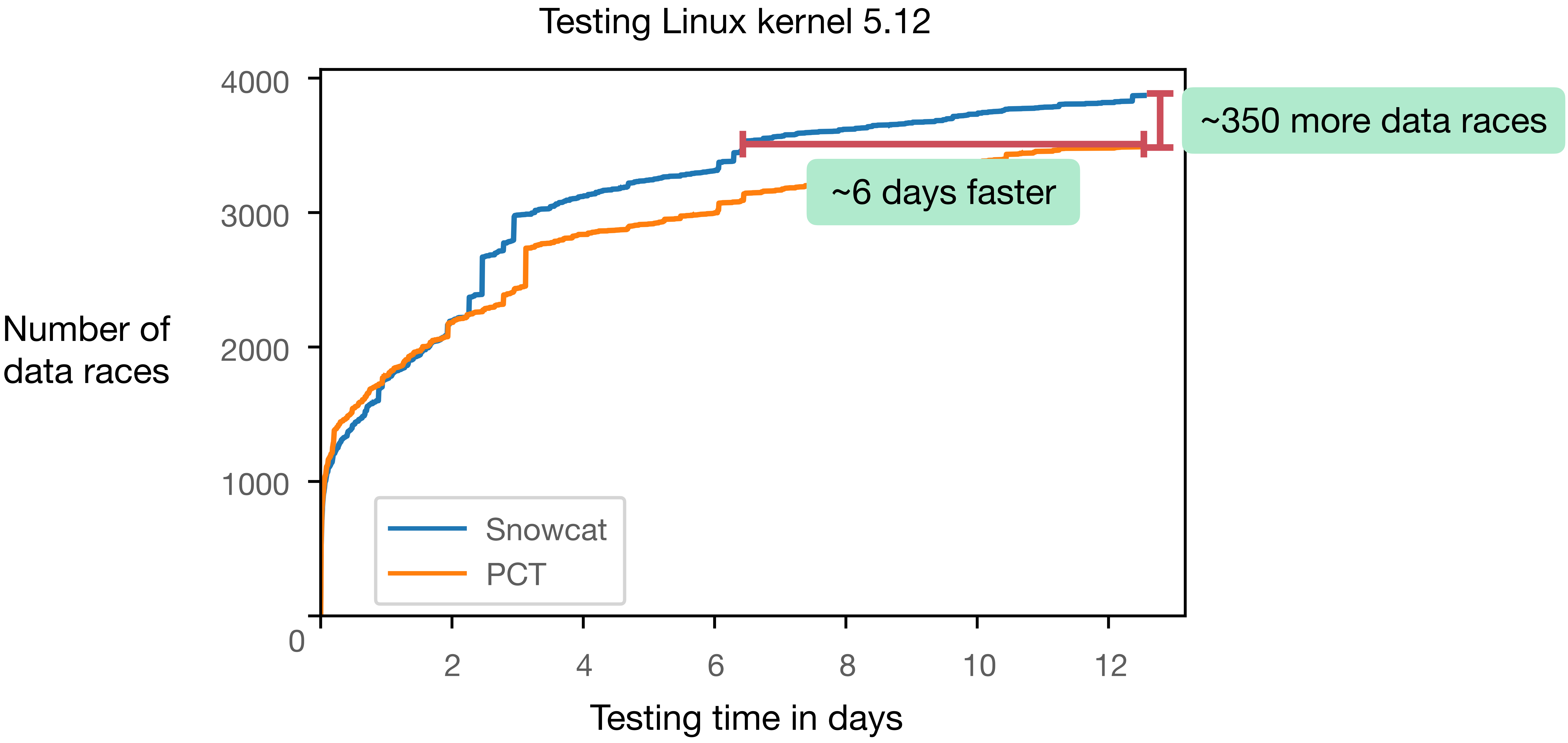
- Code block encoder
- Graph neural networks

2. Use the predictor for testing

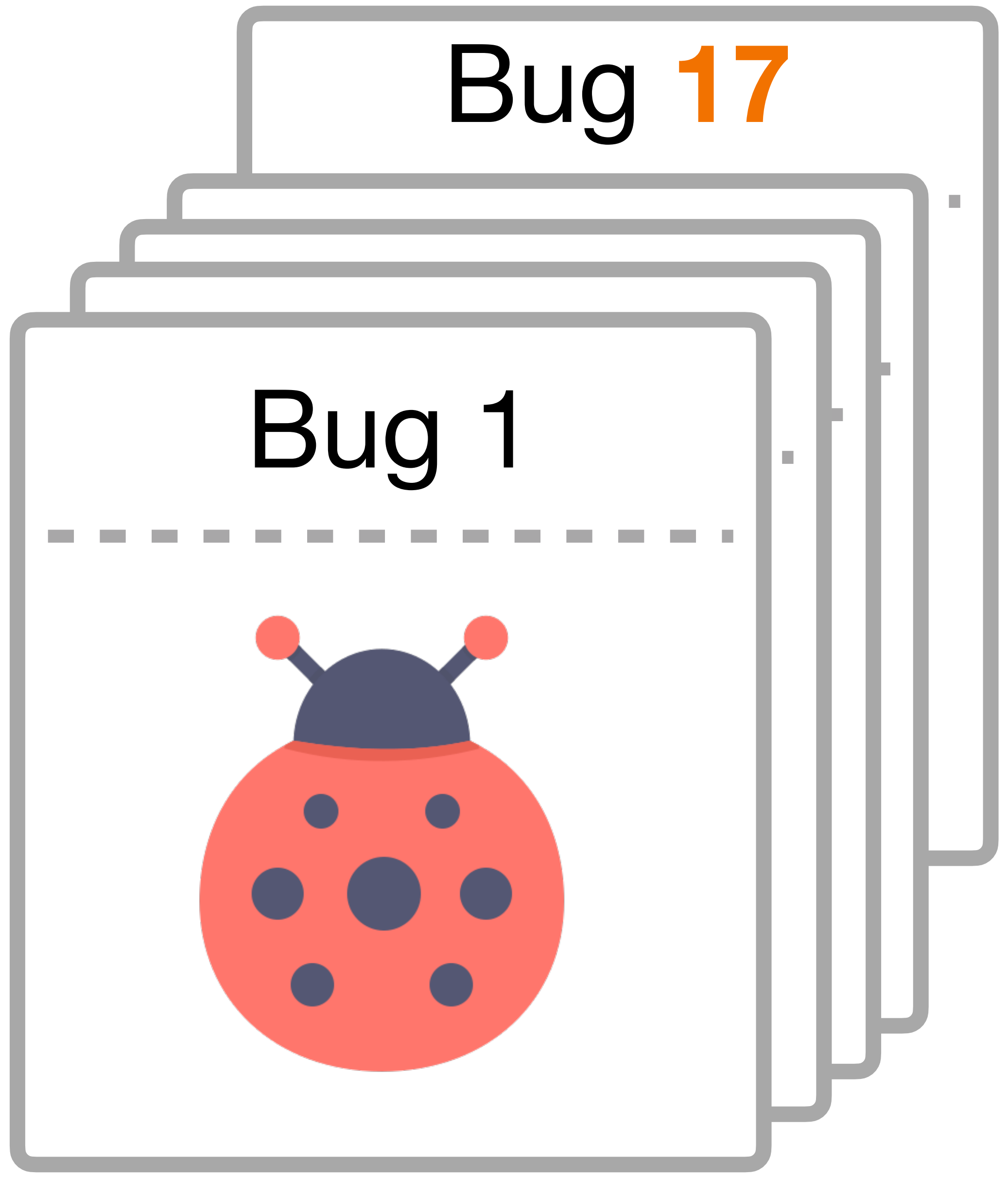
Predictor integration

- PCT [ASPLOS'10]
- Razzer [SP'19]
- Snowboard [SOSP'21]

Snowcat improves testing efficiency significantly



Snowcat is effective in finding new bugs



New concurrency bugs found in Linux kernel

13 confirmed (6 fixed)

fs/, net/, drivers/, …

Data loss, DDos, …

Existed for years (e.g, 10)

Concern about training cost

Linux kernel 5.12

Collect data, train model

10 days

Test kernel

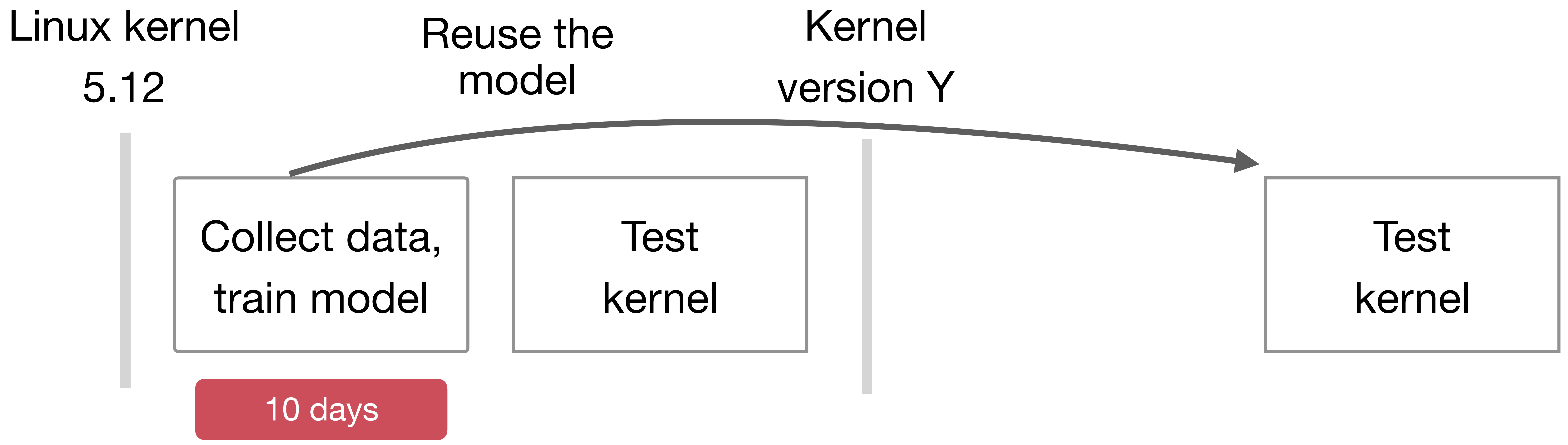
Kernel version Y

Collect data, train model

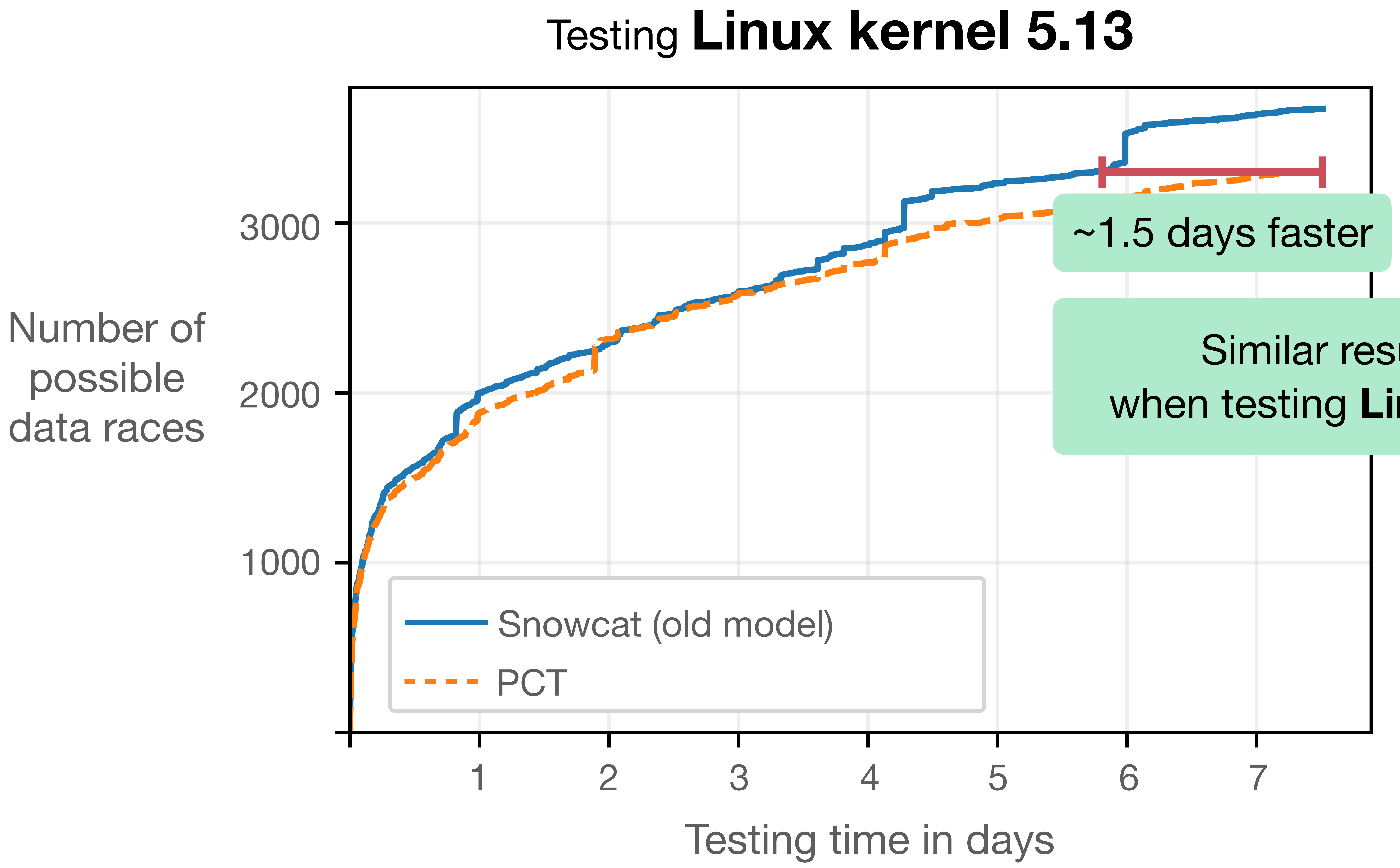
Can be reduced or even skipped

Test kernel

Reuse the model for the new kernel

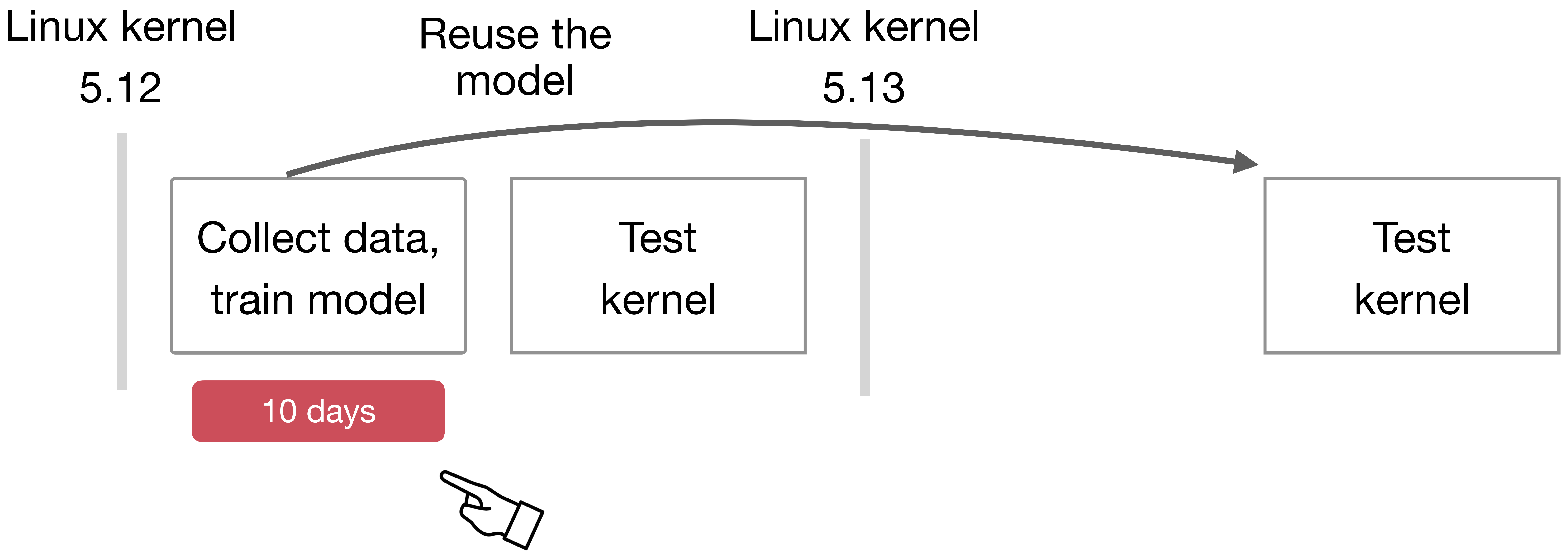


The model is still effective on the new kernel



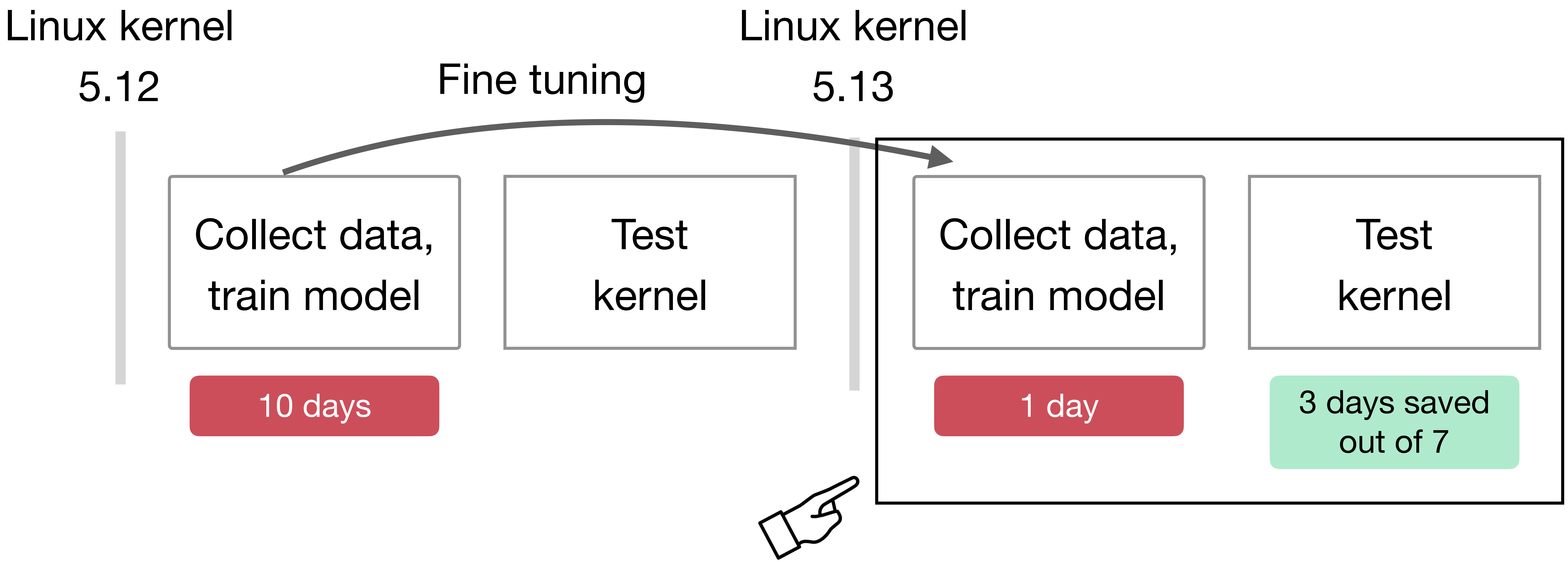
Similar results found when testing **Linux kernel 6.1**

Reuse the model for the new kernel



The foundational model works well even on new kernels

Fine tune the model for the new kernel



The fine-tuned model brings higher efficiency

You're welcome to read the paper

More details

- Test Linux 6.1
- Concurrency bug reproduction
- Existing framework integration

Key takeaway from Snowcat

Improve kernel concurrency testing using ML

Predict kernel coverage for concurrency tests

Efficient and effective

https://github.com/rssys/snowcat