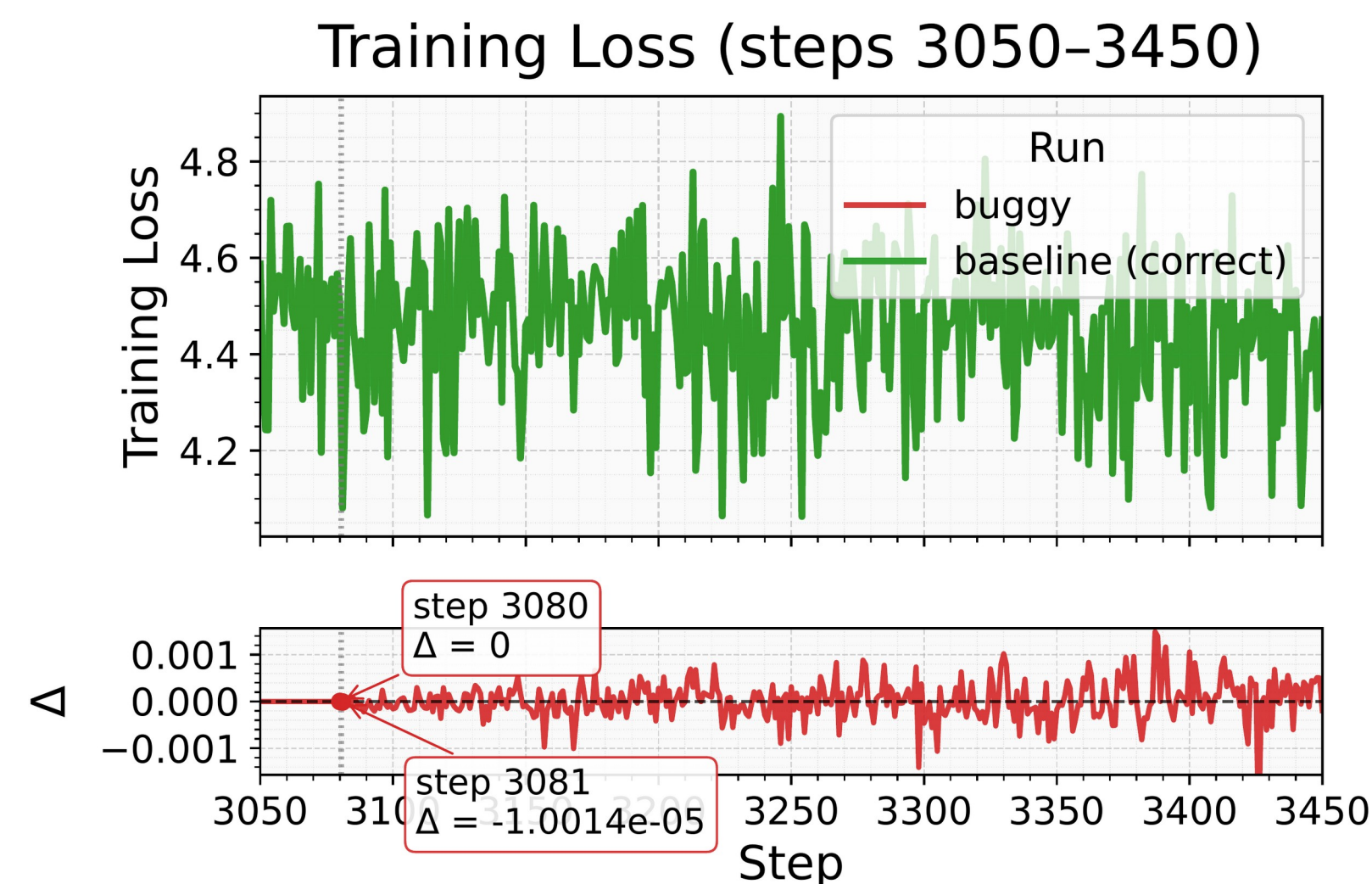


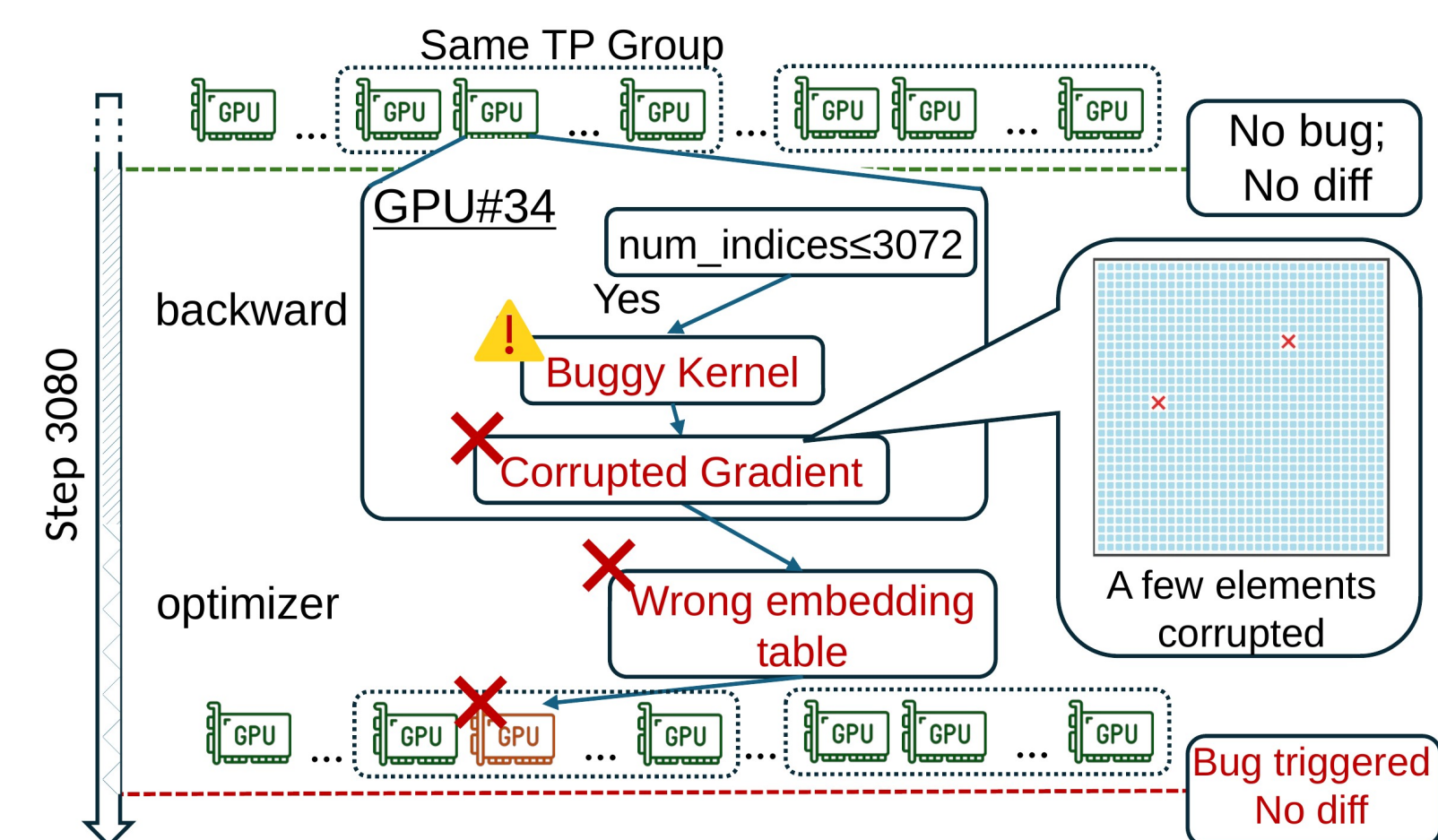
01 Loss hides the bug

Corruption can start before the loss shows a visible problem.



5 days Manual debugging
50+ reruns Loss offered little clue

A race in embedding backward
Hidden in PyTorch for 7 years



Reference run choices

Suspect	Reference
Nondeterminism	Same-stack replay
Feature / kernel	Toggle / swap it
Recent regression	Last stable commit
Cross-stack drift	Equivalent stack

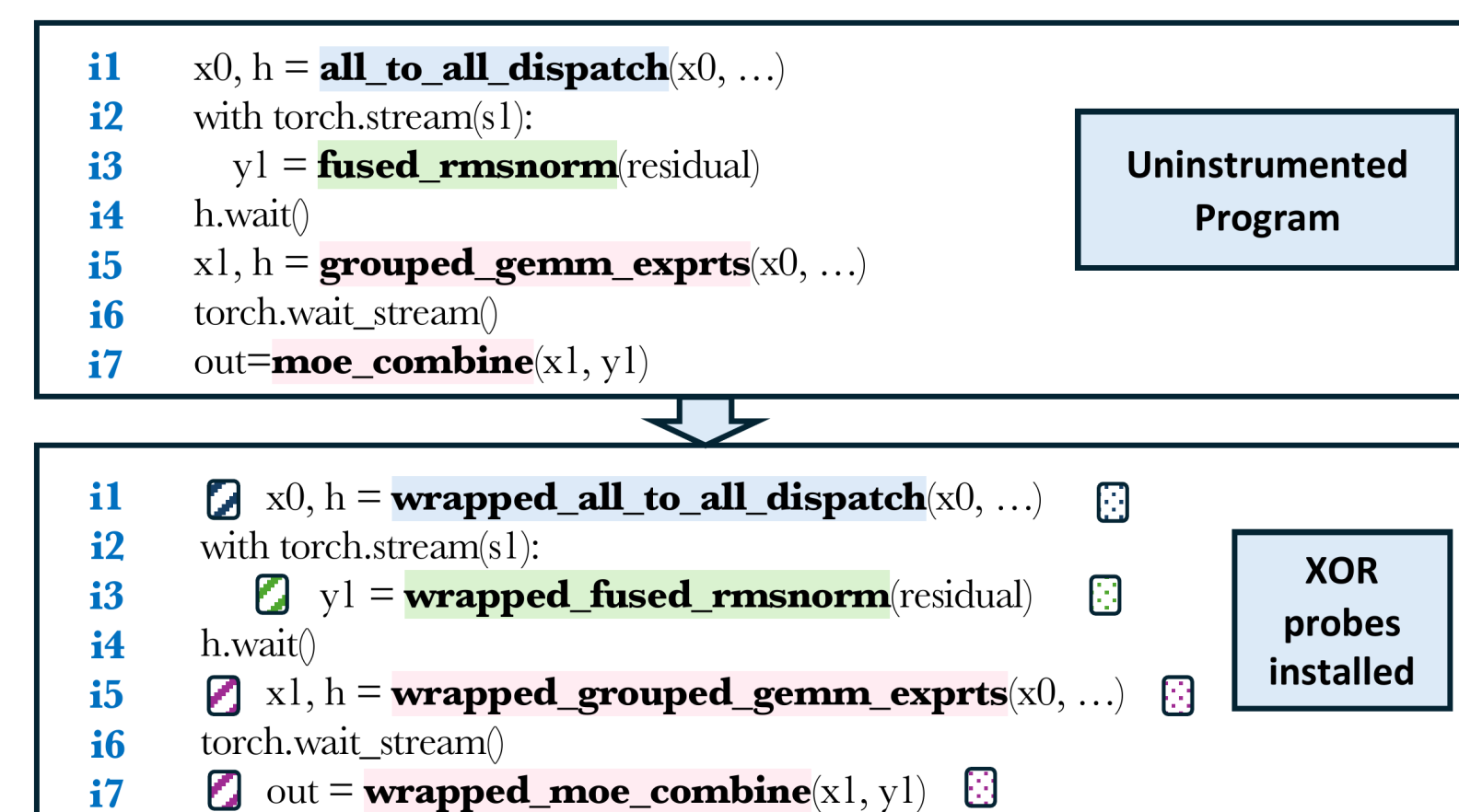
Same inputs and checkpoint.
Match RNG, precision and reduction order.

02 Instrument model Ops

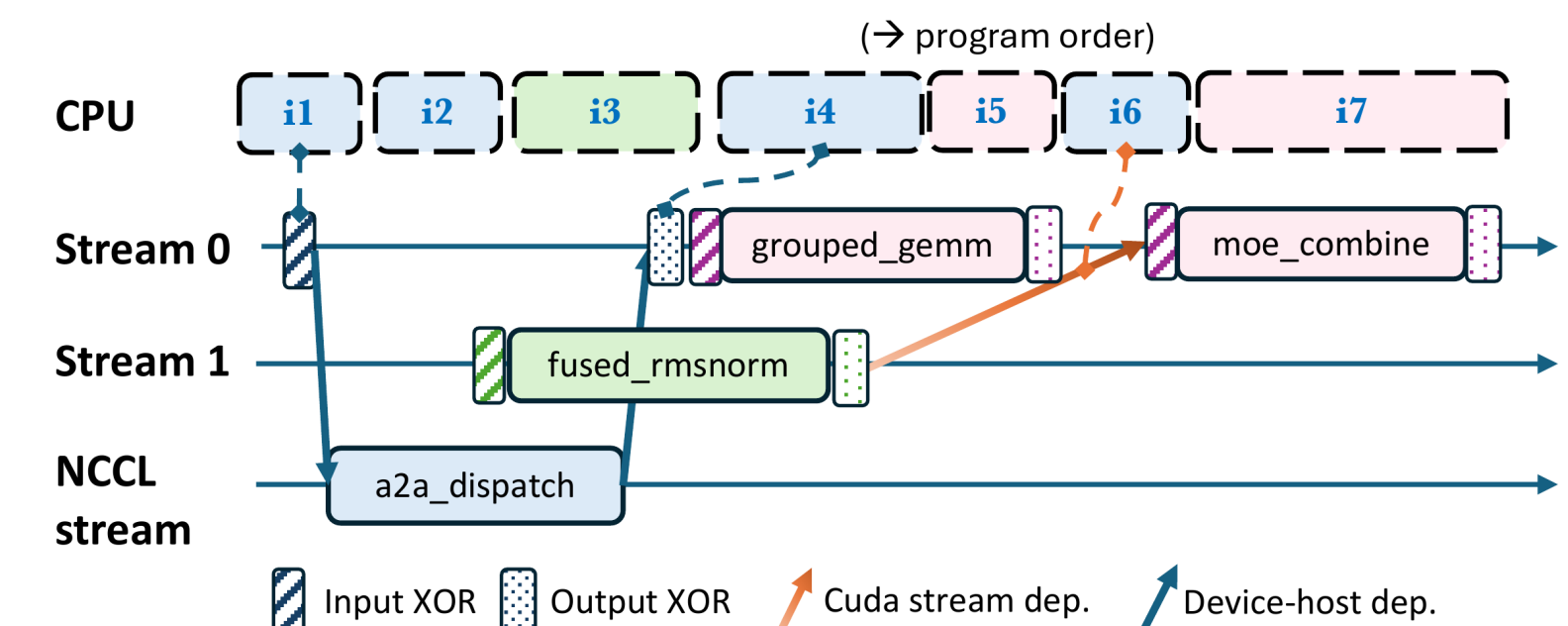
Preflight maps Python calls to GPU work.
Wrap stable model-level Op boundaries.

Wrappers preserve execution mode

AST + CST matching survives source edits.
Eager, torch.compile and CUDA Graphs.

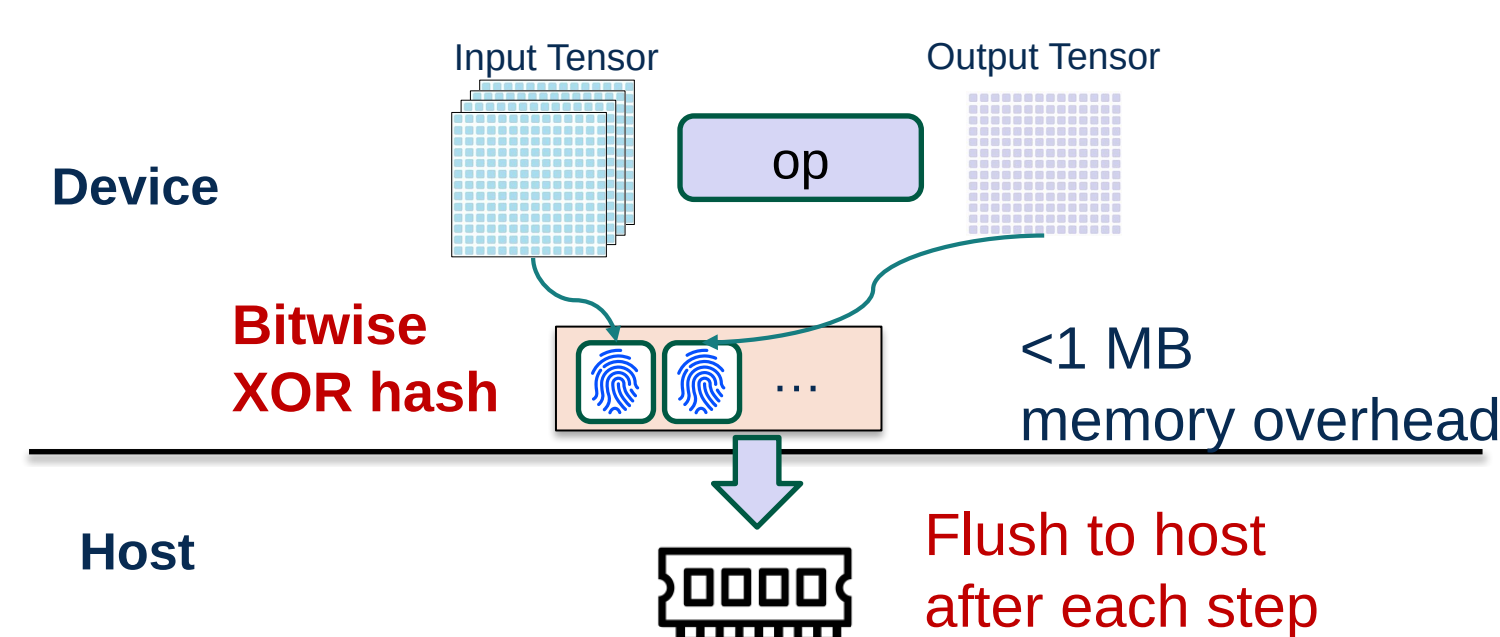


Runtime XOR hash bind to corresponding CUDA stream



Reuse `h.wait()` and `wait_stream()`.
Keep existing stream dependencies.

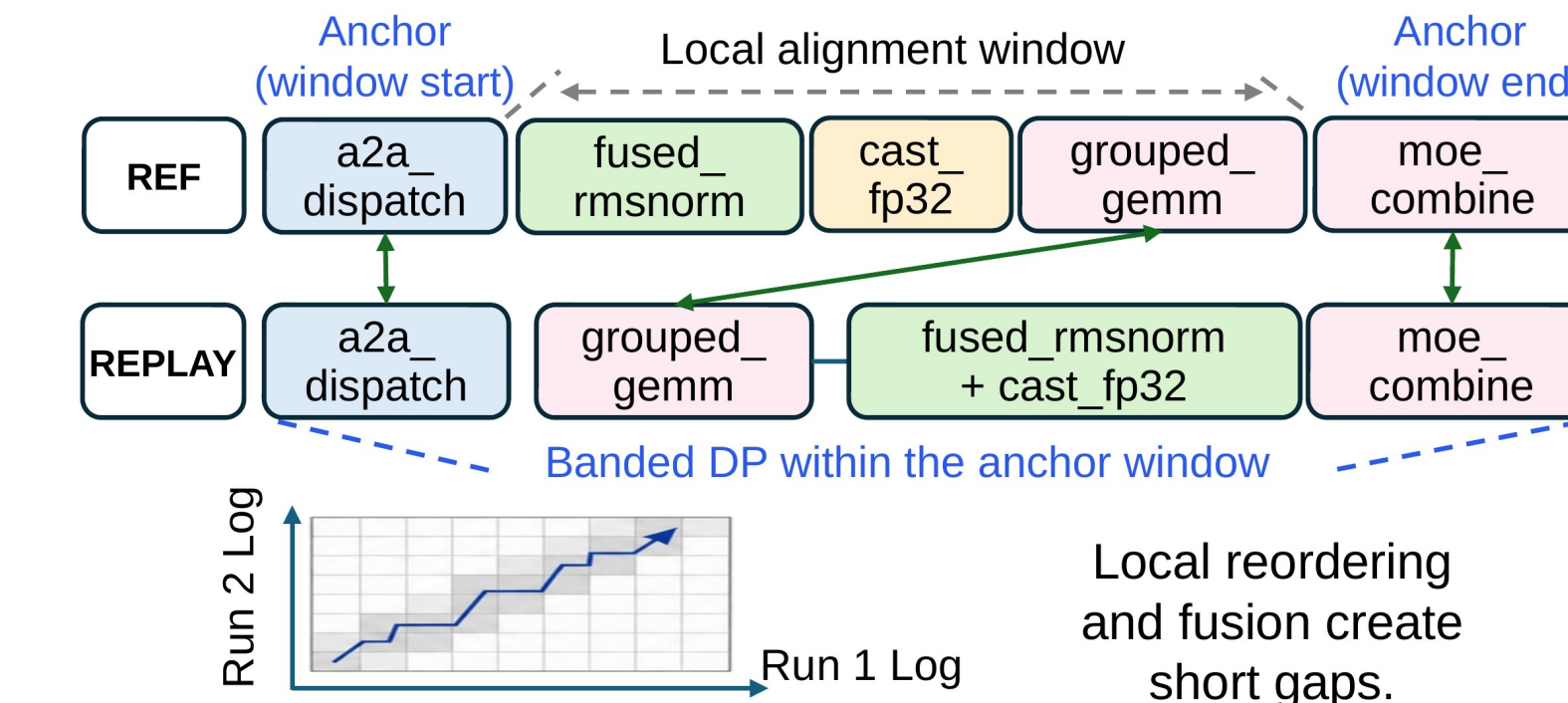
Asynchronous logging incurs low ovhd



03 Align and localize

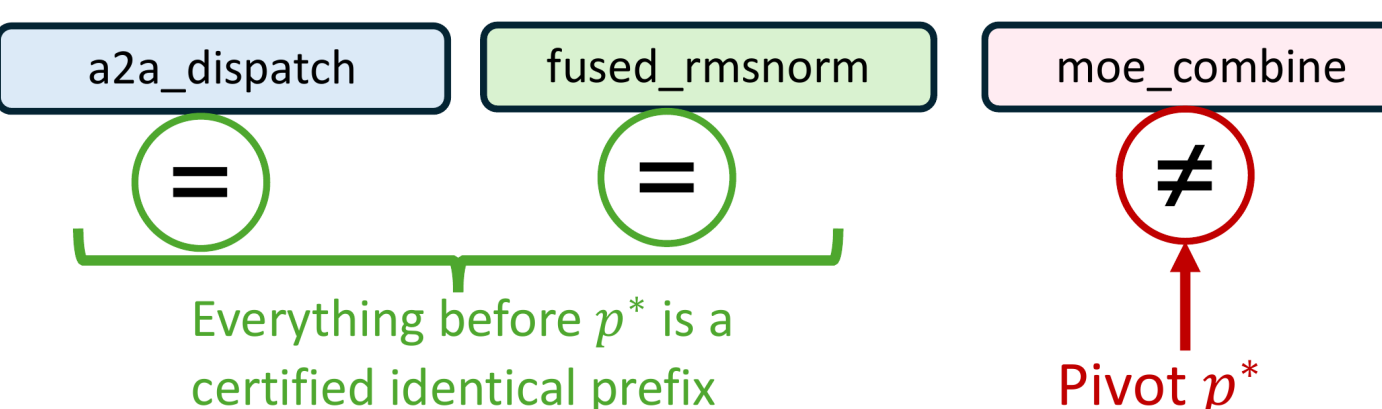
Anchors + banded DP align shared Ops across local reordering and fusion.

Schedule-tolerant alignment



Certified-prefix localization

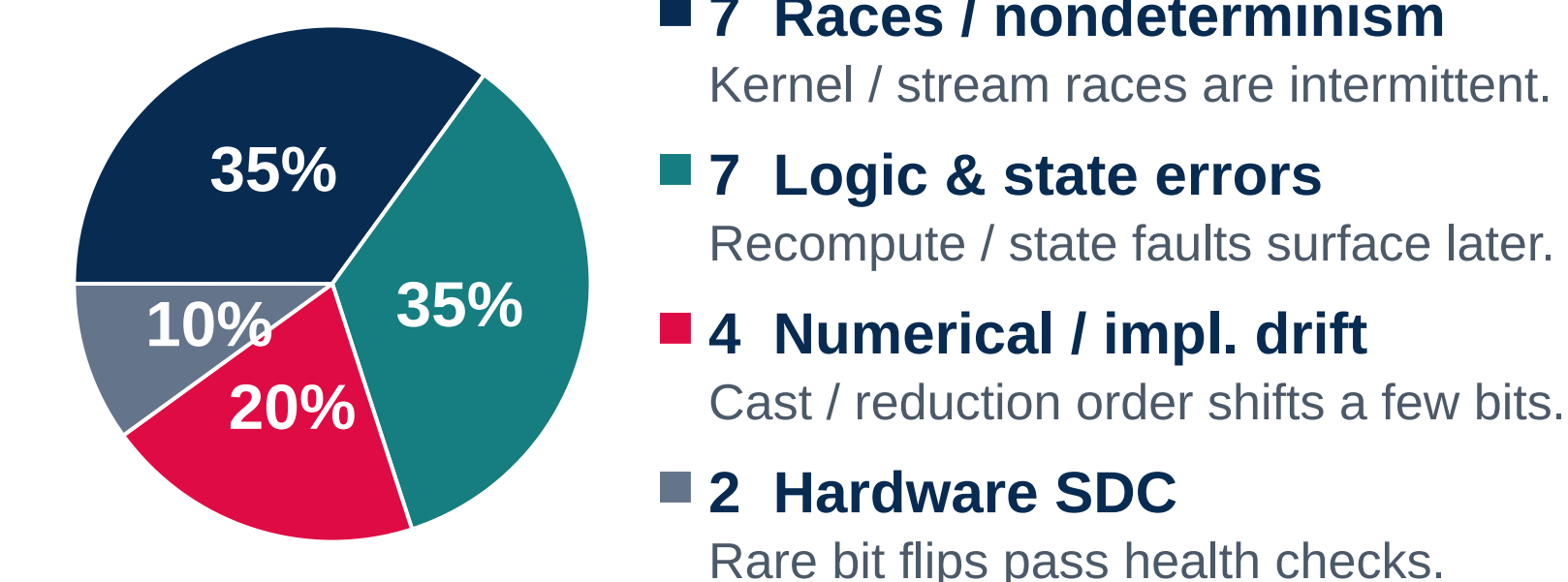
Aligned boundary pairs (in order)



First mismatch: Op + rank + source

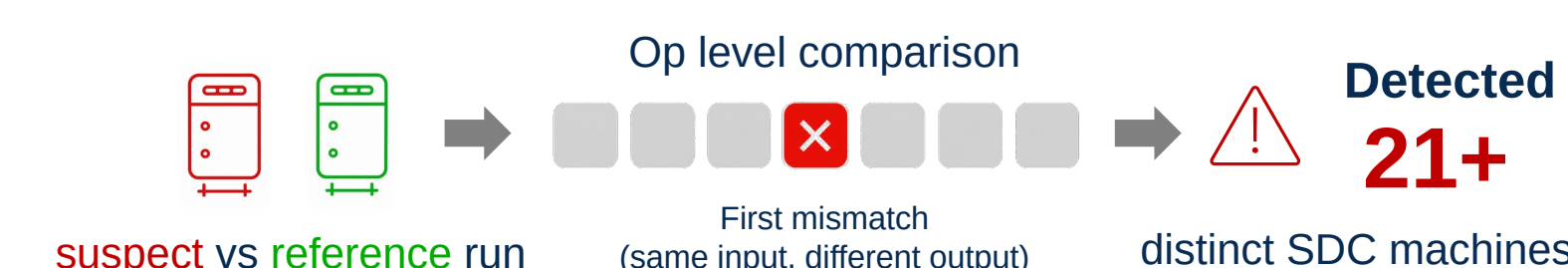
04 Production impact

Bug classes localized



Grouped by root cause from paper Table 3

Silent data corruption (SDC)



Passed health checks. Flagged in training.

Online: suspect machine. Offline: first divergent Op.

Production impact (continued)

Used by 15+ teams at ByteDance

Validated stacks and runtimes



20/20

production cases localized

8/10

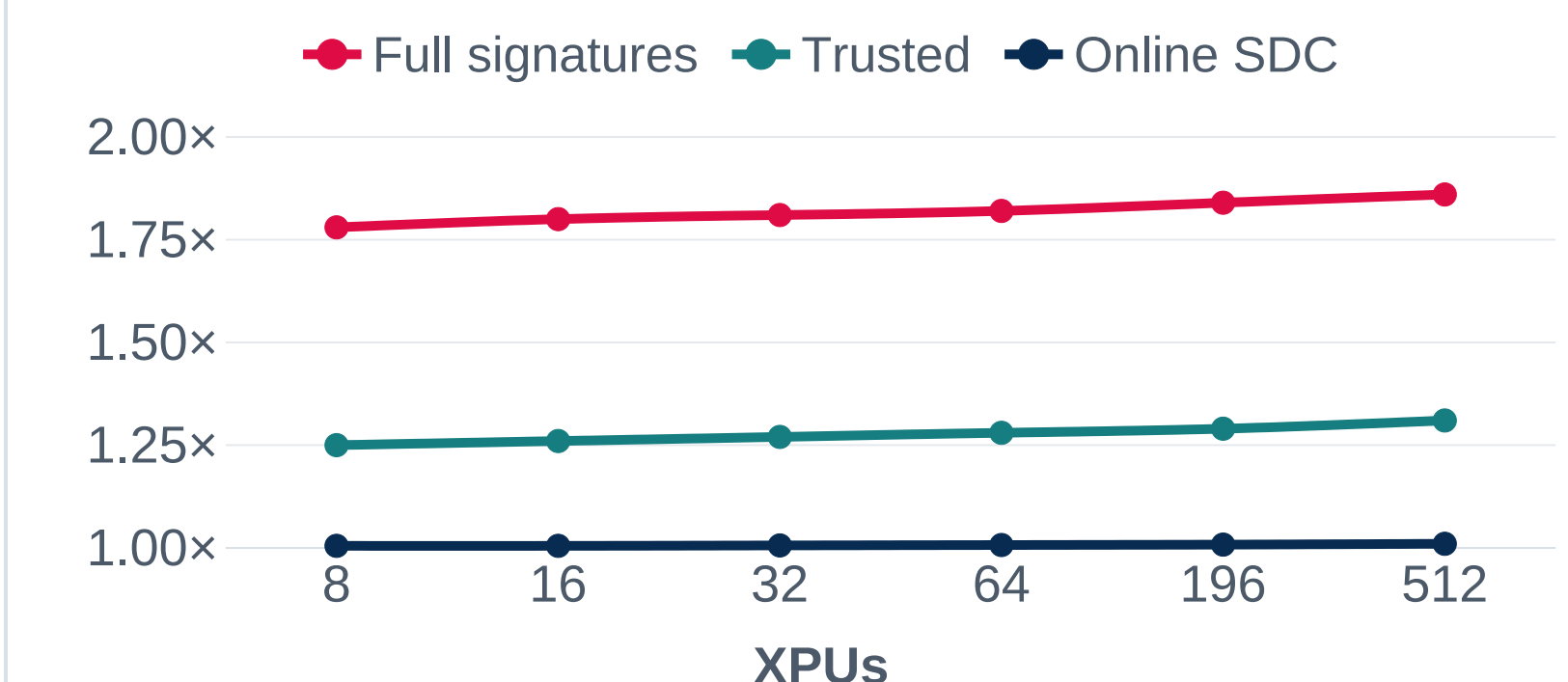
open-source cases with a precise first diff

Days of triage become minutes

Case	Manual	OpGuard
Embedding race	5 days	15 min
MoE scatter SDC	14 days	30 min
Triton barrier	4 days	10 min

Engineer-estimated OpGuard times. Paper Table 2.

Runtime cost at 8-512 XPUs



Pre-training. 1.0x = baseline.
Trusted skips signatures for trusted Ops.



Live demo

Inspect the mismatch.
Jump to source.

orderlab.io/OpGuard/