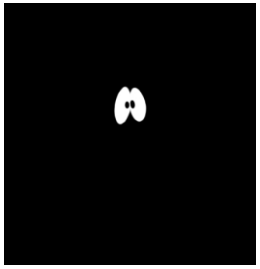
Handling Production-Run Concurrency-Bug Failures

Shan Lu
University of Chicago
Karu Sankaralingam
University of Wisconsin - Madison

Reliability is crucial

- Production-run failures are costly



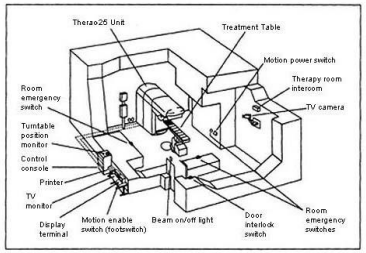


Figure 1. Typical Therac-25 facility

- Reliability affects other aspects of parallel systems

"We don't use thread-level parallelism because that is difficult to get correct"

Reliability is challenging

- Concurrency bugs
  - Synchronization problems in multi-threaded s/w

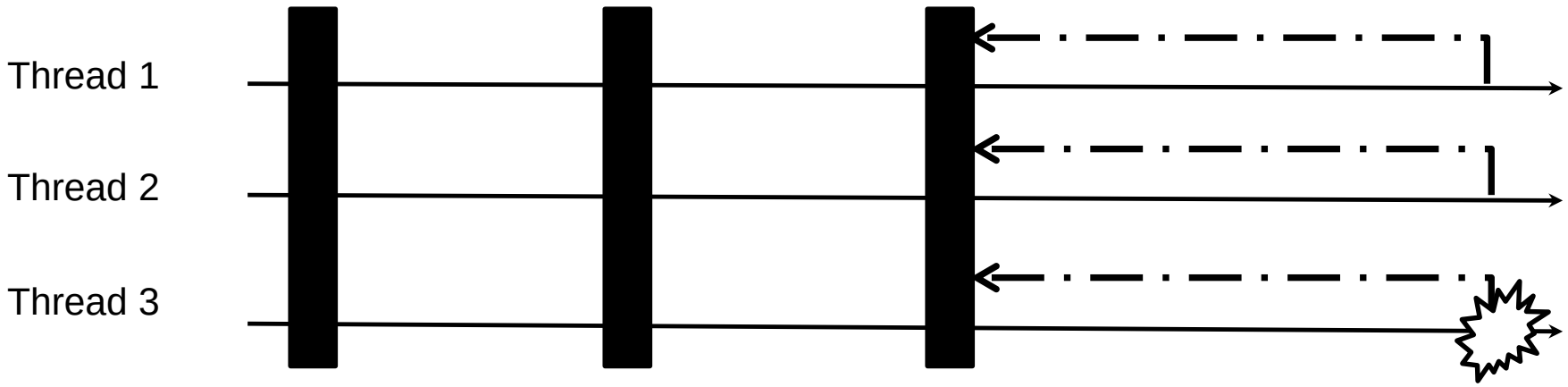
Thread 1	Thread 2
if (**proc**){	
tmp=***proc**;	**proc** = NULL;
}	

MySQL

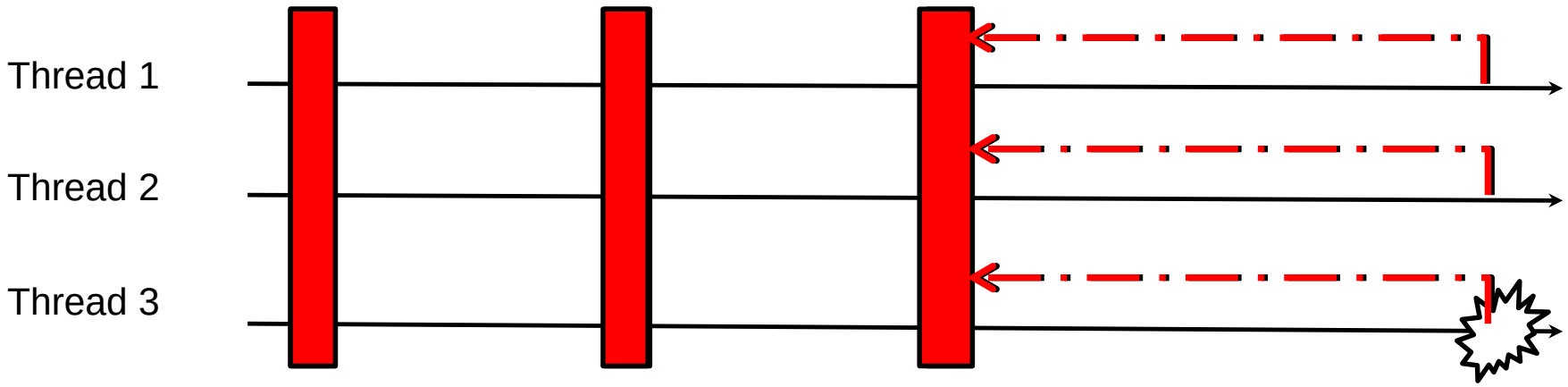
- Concurrency bugs widely exist in production
  - In-house testing is ineffective for concurrency bugs

How to handle production-run failures caused by concurrency bugs?

Rollback recovery



Problems of the traditional approach



Bug examples

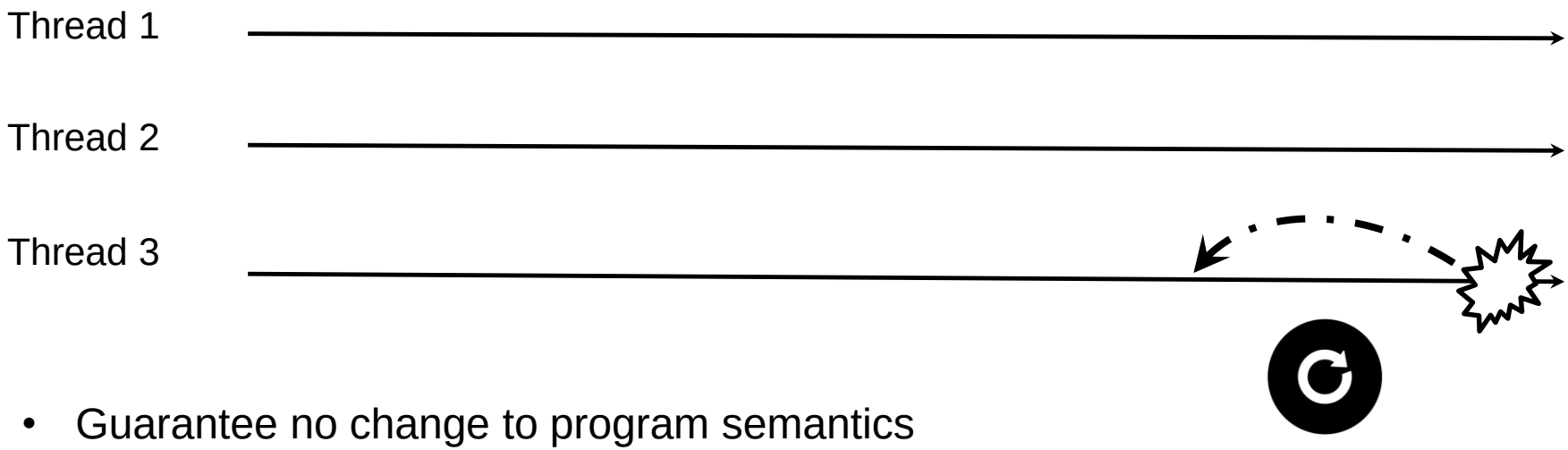
Thread 1	Thread 2
if (**proc**){	**proc** = NULL;
tmp=***proc**;	
}	

MySQL

Do we need to roll back all threads?

Do we need memory checkpoint?

ConAir system



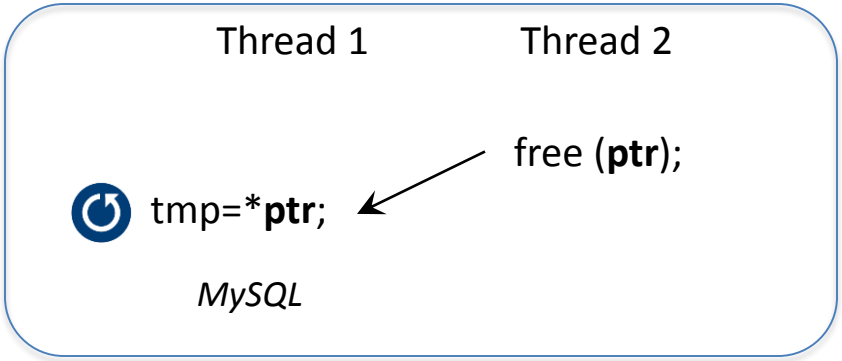
- Guarantee no change to program semantics
- No change to OS/Hardware
- No prior bug knowledge required

- Negligible overhead (<0.2%)
- Work for 16 out of 26 real-world bugs

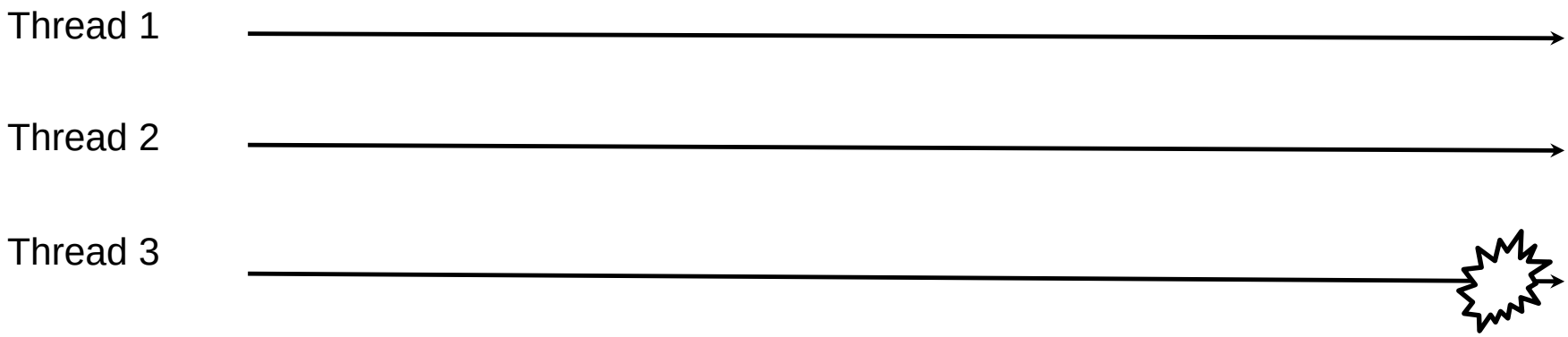
The problems

- What if the error propagation distance is long?

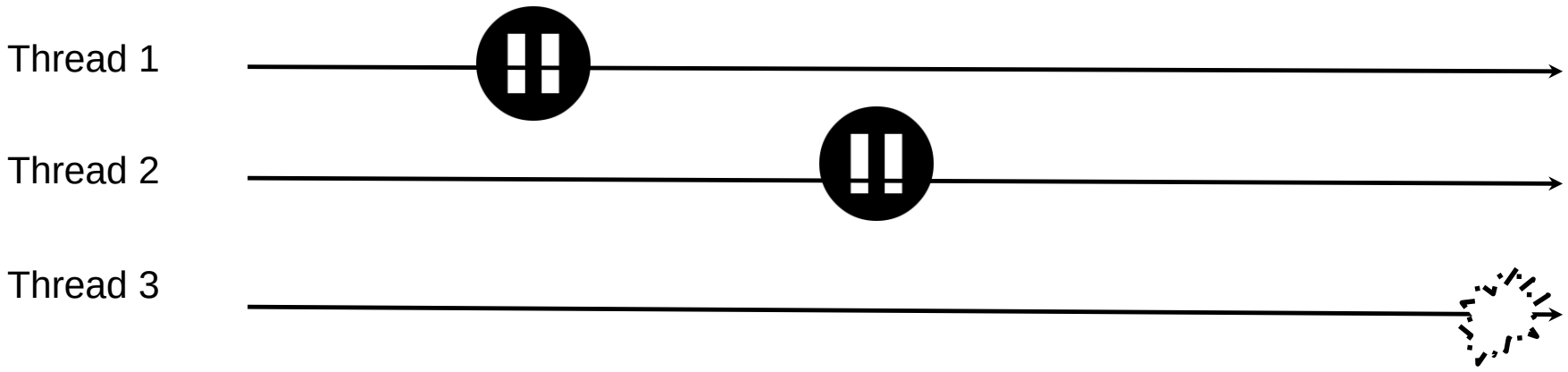
- What if the failure thread was already too slow?



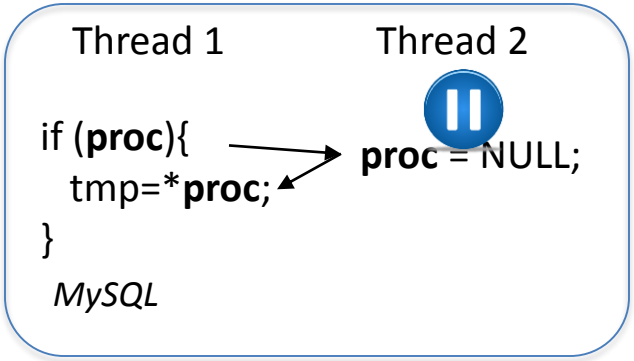
Proactive prevention

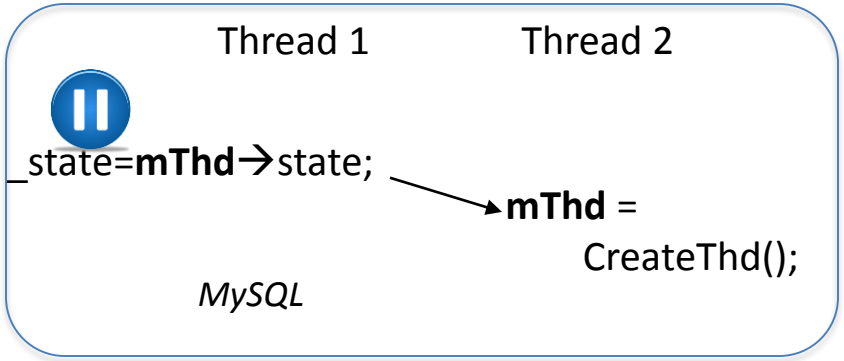


Proactive prevention



Bug examples





Do we need to perturb multiple threads?

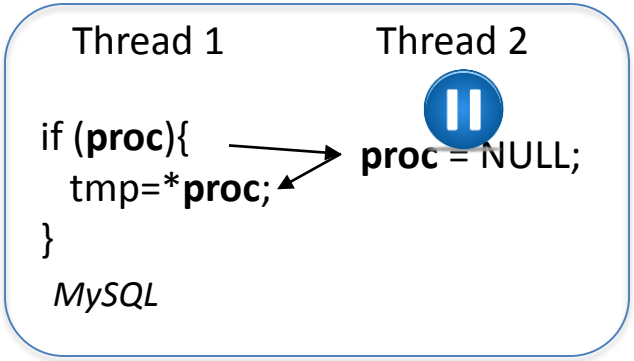
Do we perturb at random place?

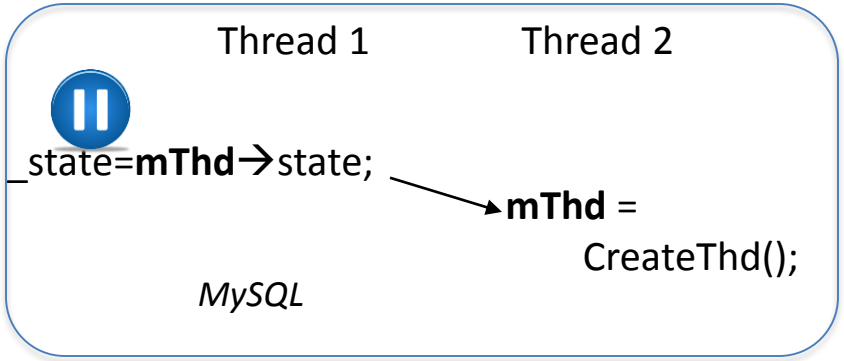
A simple and generic scheme

- The best perturbation point is

  Right before a memory access that has an incorrect/abnormal remote predecessor

AI system





- Guarantee no change to program semantics
- No change to OS/Hardware
- ~~No prior bug knowledge required~~ — Training required

- ~~Negligible overhead (<0.2%)~~ — 1% ~ 10X
- Work for ~~16 out of 26~~ real-world bugs — 35 out of 35

*AI: A Lightweight System for Tolerating Concurrency Bugs [**FSE14**]* ***ACM SIGSOFT Distinguished Paper Award***

ConAir vs. AI

	ConAir	AI
Performance	Great	Poor when there are intensive shared-memory accesses
Functionality	Poor when failure thread is too slow Poor when error propagation is long	Great, but require training

- Can we combine ConAir and AI?

Summary & Other efforts

- Reactive approach to production-run failures

  *ConAir: Featherweight Concurrency Bug Recovery Via Single-Threaded Idempotent Execution [**ASPLOS13**]*

- Proactive approach to production-run failures

  *AI: A Lightweight System for Tolerating Concurrency Bugs [**FSE14**]*

  ACM SIGSOFT Distinguished Paper Award

- How much can developers contribute?

  *What change history tells us about thread synchronization [**FSE15**]*

- How much can hardware contribute?

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY of WISCONSIN MADISON

Thank XPS!