Durinn: Adversarial Memory and Thread Interleaving for Detecting Durable Linearizability Bugs

Xinwei Fu*, Dongyoon Lee+, Changwoo Min*

* VIRGINIA TECH

+ Stony Brook University

Summary

- Crash-consistent software without paying storage overhead
- Writing crash-consistent programs is error-prone

- NVM Correctness Condition: Durable Linearizability

Durinn
- The first Durable Linearizability checker
- Three Durable Linearizability bug patterns
- Adversarial crash state and thread interleaving construction
- Likely-Linearization Point inference
- Detected 27 (15 new) bugs

Outline

- ***<u>Introduction</u>***
- Durinn
- Evaluation
- Conclusion

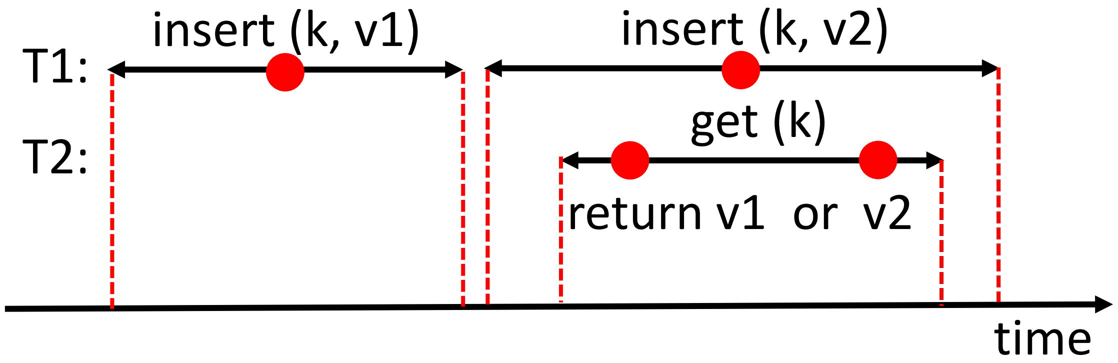
NVM Correctness Condition: Durable Linearizability

Durable Linearizability requires:

- (C1) without a crash, all operations are *Linearizable*

Linearizability requires that all operations:

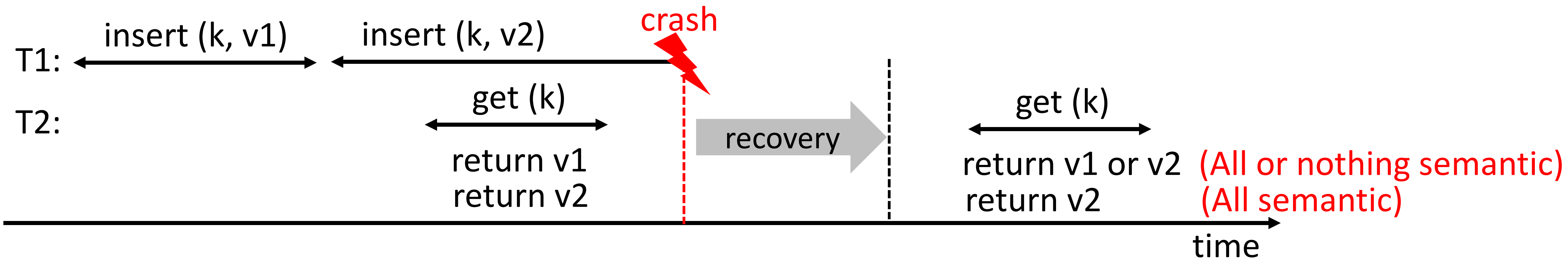
- take effect instantaneously at a ***program point*** (*Linearization Point*)
- and that point is between the operation begin and end



NVM Correctness Condition: Durable Linearizability

Durable Linearizability requires:

- (C1) without a crash, all operations are linearizable
- (C2) completed operations before a crash → All semantic
- (C3) incomplete operations upon a crash → All or nothing semantic

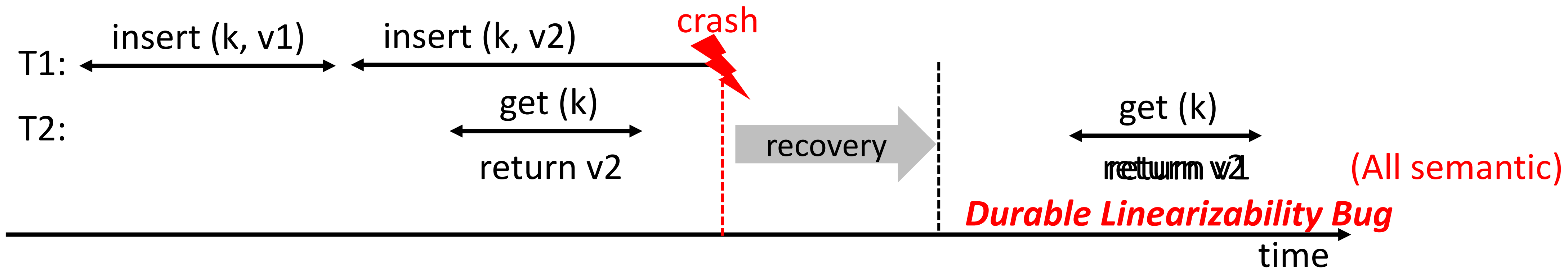


NVM Correctness Condition: Durable Linearizability

Durable Linearizability describes correct operation behaviors:

- Crash state
- Thread interleaving

Any incorrect operation behavior leads to a Durable Linearizability bug.



Our Contributions

Existing Solutions

- Linearizability testing tools
- NVM-specific crash-consistency bug detectors

Durinn

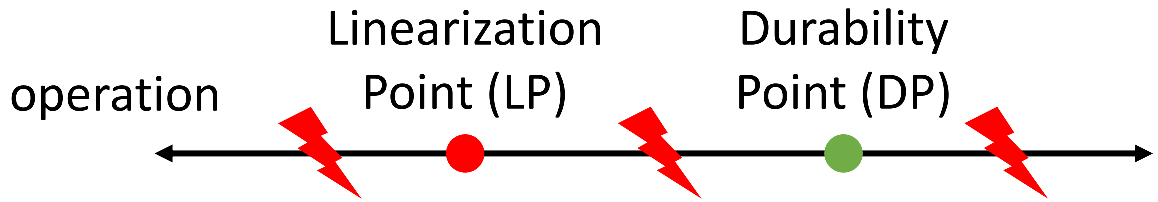
- Three Durable Linearizability bug patterns
- Adversarial NVM State and Thread Interleaving Construction
- Likely-Linearization Point Inference

Outline

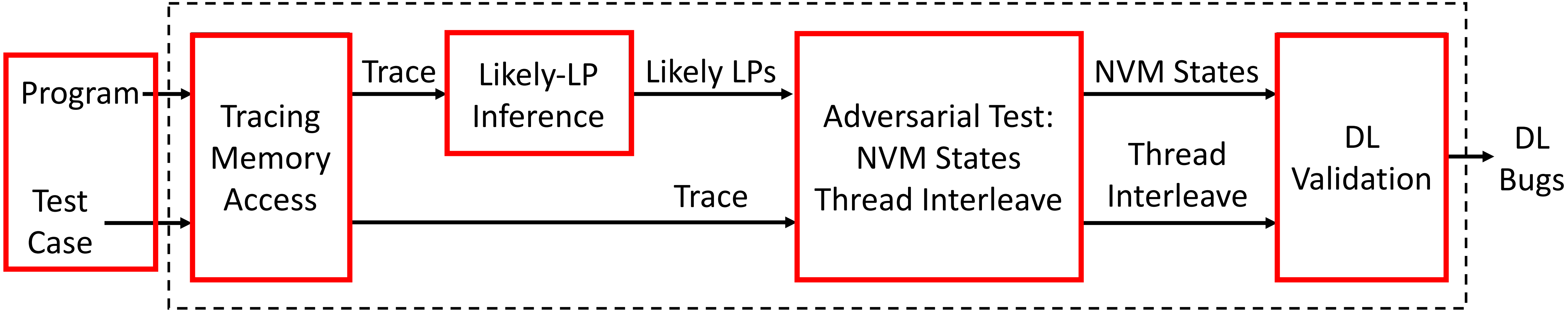
- Introduction
- ***<u>Durinn</u>***
- Evaluation
- Conclusion

Durinn Overview

- Linearization Point → understand operation behaviors



- **Key idea 1:** three durable linearizability bug patterns
- **Key idea 2:** adversarial test for both crash state and thread interleaving



Outline

- Introduction
- Durinn
  - ***Durable Linearizability Bugs and Adversarial Testing***
  - Likely-Linearization Point Inference
- Evaluation
- Conclusion

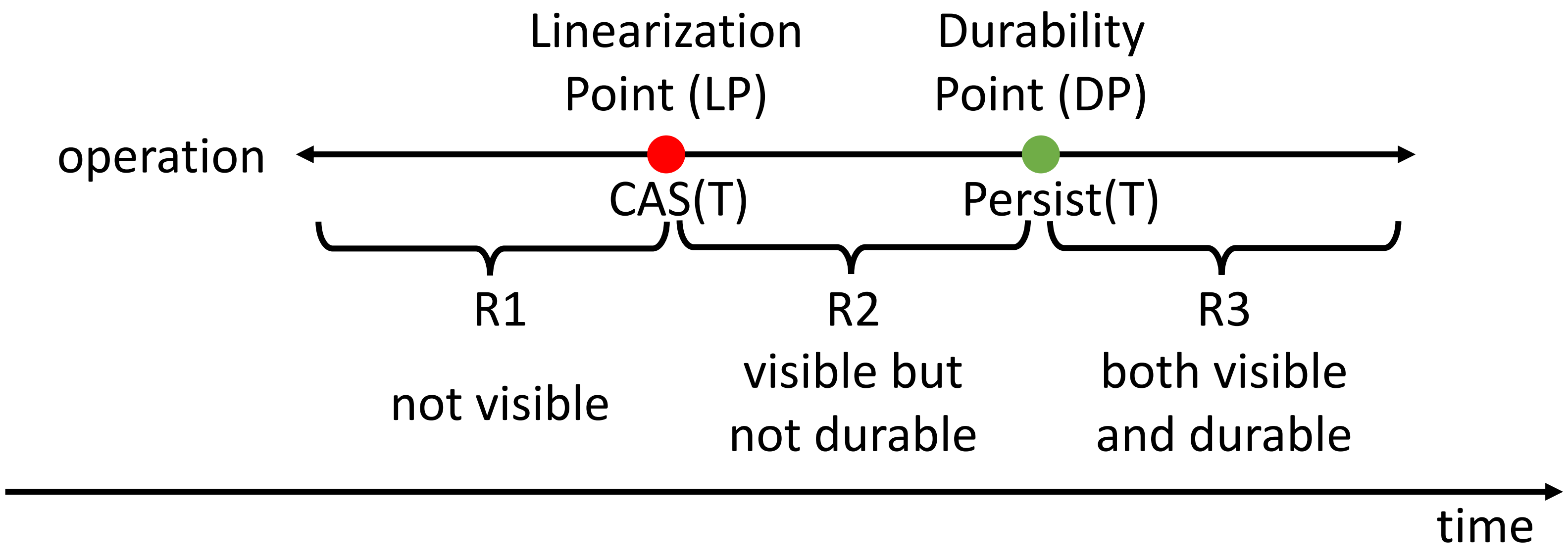
The gap between LP and DP

Linearization Point

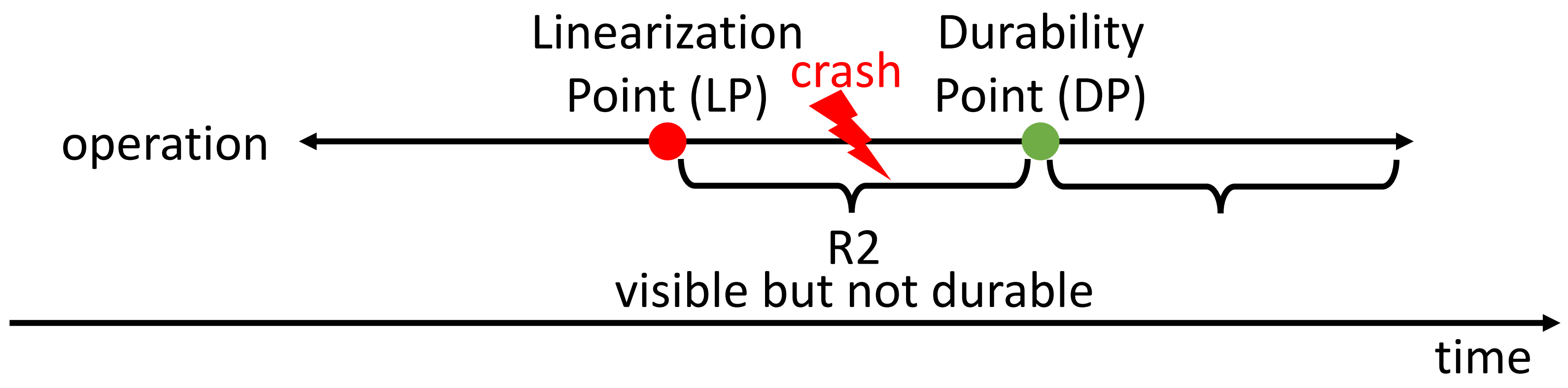
a program point where an operation takes effect and its effects become visible

Durability Point

a program point where the effect becomes persisted

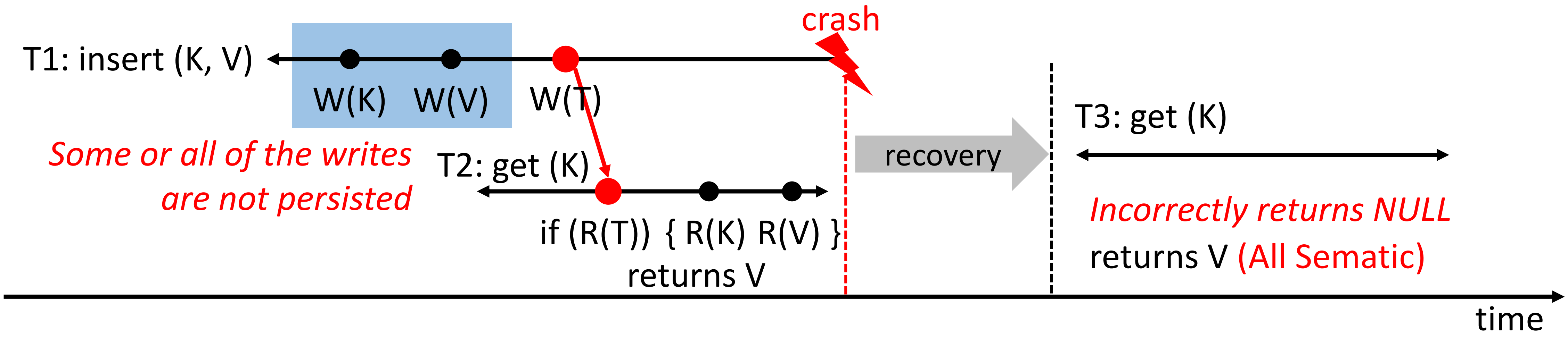


DL3 Bug: A Visible-But-Not-Durable Bug



Correctness condition:

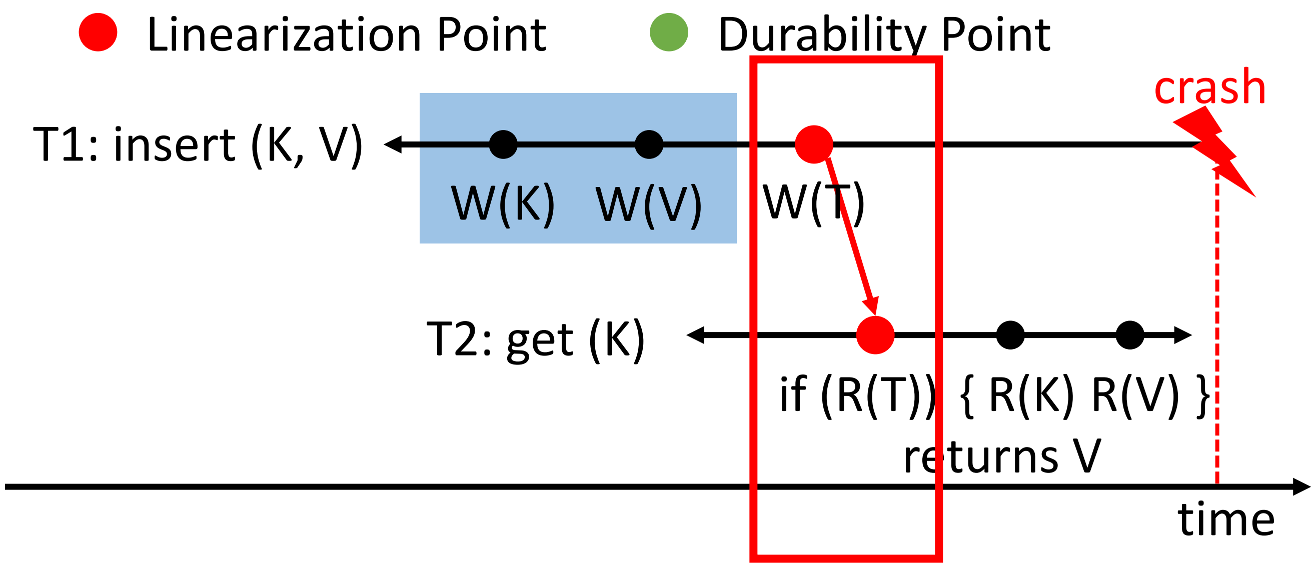
A crash between LP and DP, if the effect has been observed before crash, the operation should preserve All Semantic.



Adversarial test for DL3 (Visible-But-Not-Durable) Bug

Correctness condition:

A crash between LP and DP, if the effect has been observed before crash, the operation should preserve All Semantic.

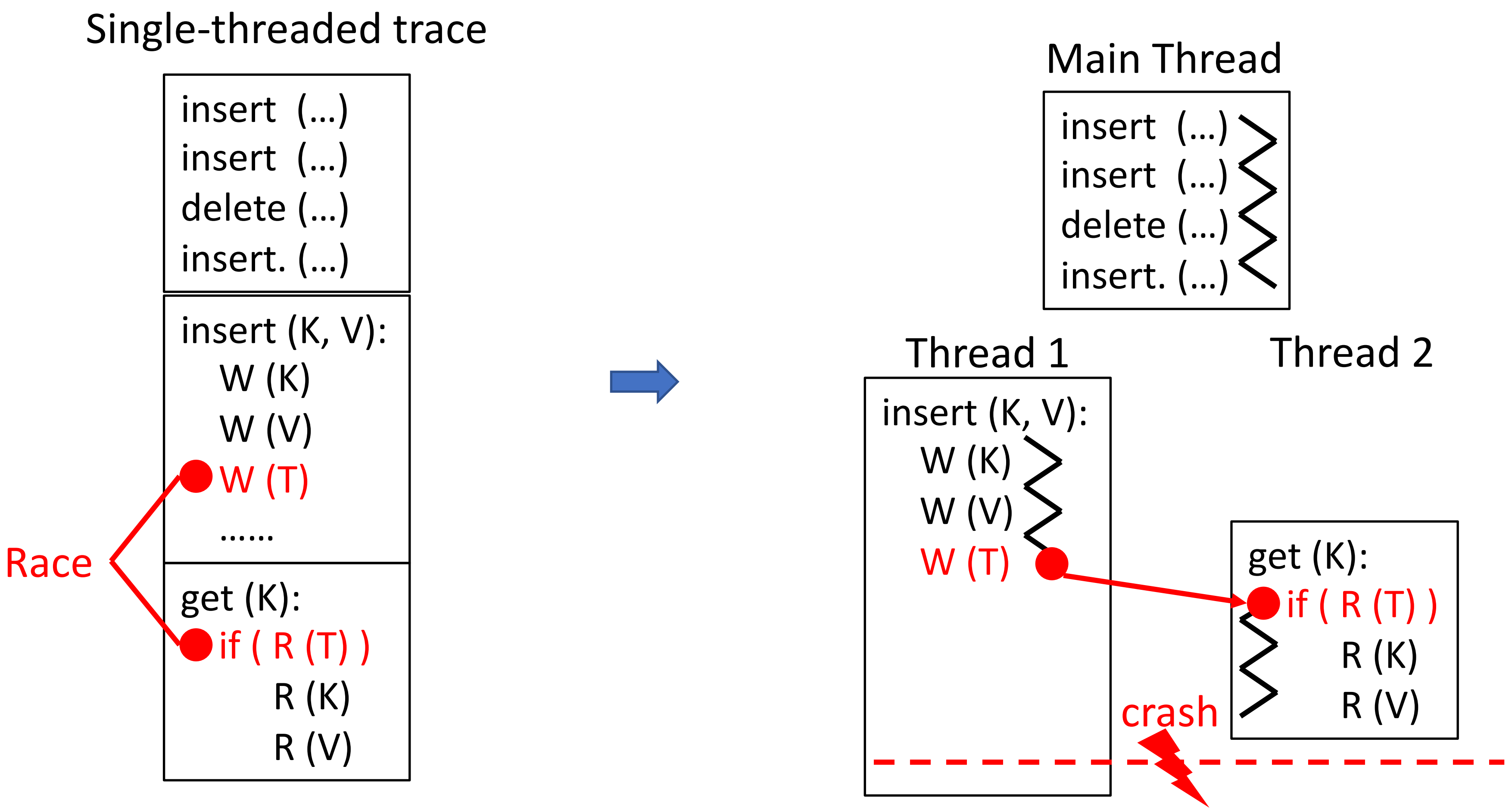


All legal crash states

K	V	
Persisted	Persisted	
Persisted	Unpersisted	
Unpersisted	Persisted	
Unpersisted	Unpersisted	*Worst case*

- A pair of racy operations
- A specific thread interleaving

Adversarial test for DL3 Bug



Outline

- Introduction
- Durinn
  - Durable Linearizability Bugs and Adversarial Testing
  - ***Likely-Linearization Point Inference***
- Evaluation
- Conclusion

Likely-Linearization Point Inference

(1) Atomic Instruction

Lock-free Insert

```
......
Atomic Inst
● CAS (T)
......
```

● Linearization Point

(2) Guarded-Protection Pattern

Insert (K, V)

```
W(K)
W(V)
Persist(KV)
Set guardian
● W(T)
Persist(T)
```

Get (K)

```
Read guardian
● if ( R(T) ) {
      R(K)
      R(V)
  }
```

(3) Publish-after-Initialization

Insert

```
node = malloc()
Initialization
node.key = k
node.val  = v
......
Publish
● prev.next = node
```

No LPs.

Outline

- Introduction
- Durinn
- ***<u>Evaluation</u>***
- Conclusion

Evaluation

Tested Applications:

- 13 concurrent NVM data structures
  - Array, queue, linked list, skip list, hashtable, radix tree, B+tree and trie
- Low-level persistence primitives and high-level persistence transactions
- Lock-based and lock-free
- 1000 operations generated by AFL++ fuzzer

Evaluation Questions:

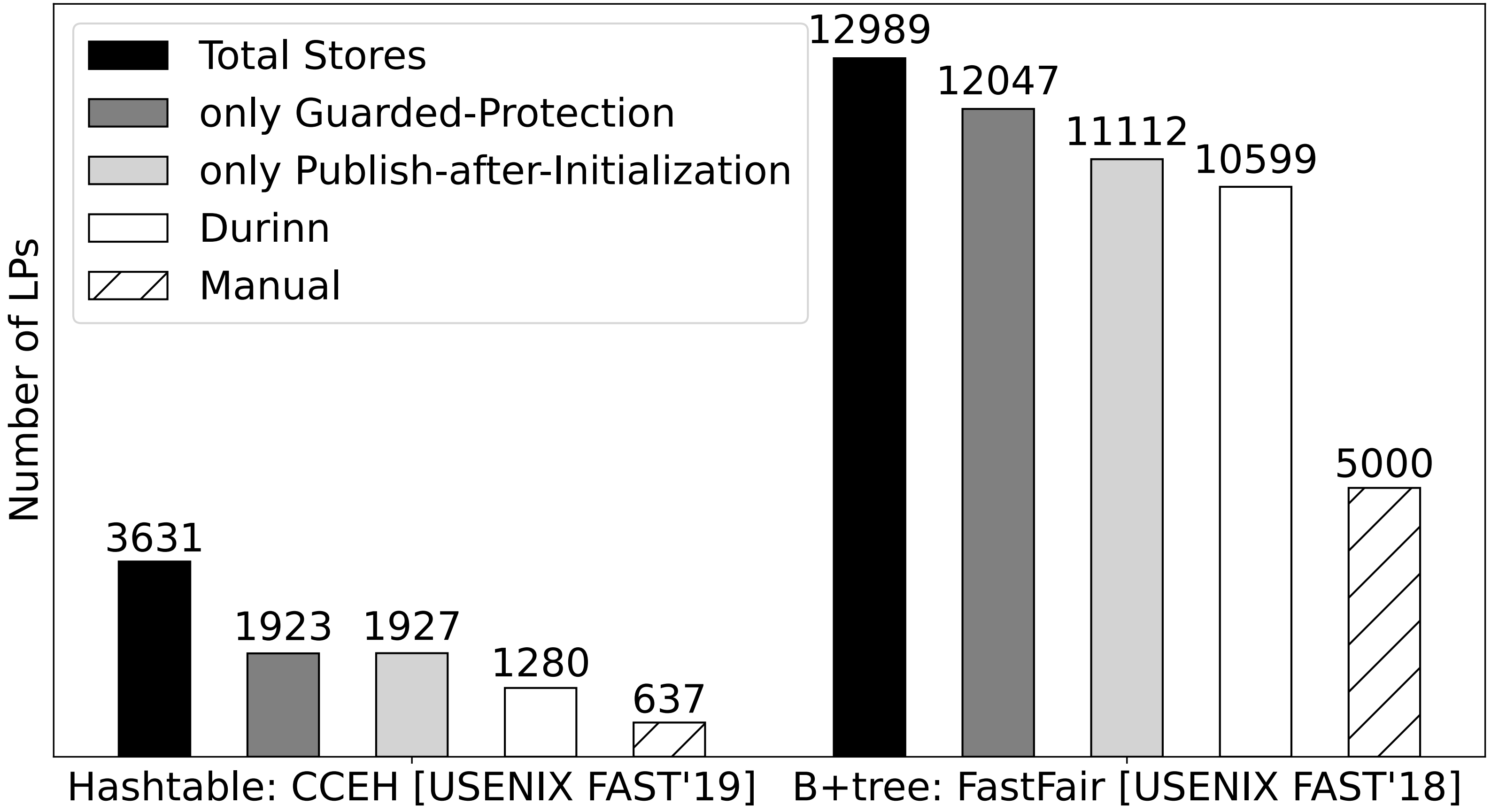
- Can Durinn detect new bugs?
- How effective and sound is Durinn's likely-LP technique?
- Does Durinn outperform the state-of-the-art?

Detected DL bugs

Detected 10 DL1 bugs, 7 DL2 bugs, and 10 DL3 bugs.

Name (Total #Bugs)	Bug ID	New	Confirm	Code	Type	Description	Impact	Fix strategy
P-LF-BST (1)	1	✓	✓	BSTAravindTraverse.h:331	DL1	Missing persistence primitives	Points to garbage	add persistence primitives
P-LF-Hash (1)	2	✓	✓	ListTraverse.h:212	DL1	Missing persistence primitives	Points to garbage	add persistence primitives
P-LF-List (1)	3	✓	✓	ListTraverse.h:212	DL1	Missing persistence primitives	Points to garbage	add persistence primitives
P-LF-Skiplist(1)	4	✓	✓	SkiplistTraverse.h:218	DL1	Missing persistence primitives	Points to garbage	add persistence primitives
P-LF-Queue(1)	5	✓	✓	DurableQueue.h:L74	DL1	Missing persistence primitives	Points to garbage	add persistence primitives
CCEH (2)	6	✓		CCEH_MSB.cpp:280	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	7		✓	CCEH_MSB.cpp:103	DL2	Atomicity in rehashing	Unable to recover	inconsistency-recoverable design
FAST-FAIR (5)	8	✓	✓	btree.h:955,979	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	9	✓	✓	btree.h:955,1007	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	10		✓	btree.h:224	DL1	Missing persistence primitives	Lost key-value	add persistence primitives
	11		✓	btree.h:213	DL2	Partial inconsistency is never recovered	unable to recover	inconsistency-recoverable design
	12		✓	btree.h:576	DL2	Atomicity in node splitting	unable to recover	logging/transaction
P-ART (4)	13	✓		Tree.cpp:35,258	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	14	✓		Tree.cpp:35,384	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	15		✓	N16.cpp:15	DL2	Atomicity between metadata and key-value	Unable to recover	inconsistency-tolerable design [15]
	16		✓	N4.cpp:17	DL2	Atomicity between metadata and key-value	Unable to recover	inconsistency-tolerable design [15]
P-CLHT (3)	17	✓		clht_lb_res.c:315,370	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	18	✓		clht_lb_res.c:315,468	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	19		✓	clht_lb_res.c:166	DL1	Missing persistence primitives	Lost key-value	add persistence primitives [14]
P-HOT (4)	20	✓		HOTRowex.hpp:61,84	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	21		✓	TwoEntriesNode.hpp:30	DL1	Missing persistence primitives	Points to garbage	add persistence primitives [14]
	22		✓	HOTRowexNode.hpp:315	DL1	Missing persistence primitives	Points to garbage	add persistence primitives [14]
	23		✓	HOTRowex.hpp:270	DL1	Missing persistence primitives	Points to garbage	add persistence primitives [14]
P-Masstree (3)	24	✓		masstree.h:1837,744	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	25	✓		masstree.h:1837,941	DL3	Incorrect concurrency control	Lost key-value	fix concurrency control/help persist
	26		✓	masstree.h:1378	DL2	Atomicity in node splitting	Unable to recover	logging/transaction
pmdk-array (1)	27		✓	array.c:486	DL2	Atomicity between metadata and data	Unable to recover	logging/transaxtion

Effectiveness and soundness of Likely-Linearization Point Inference

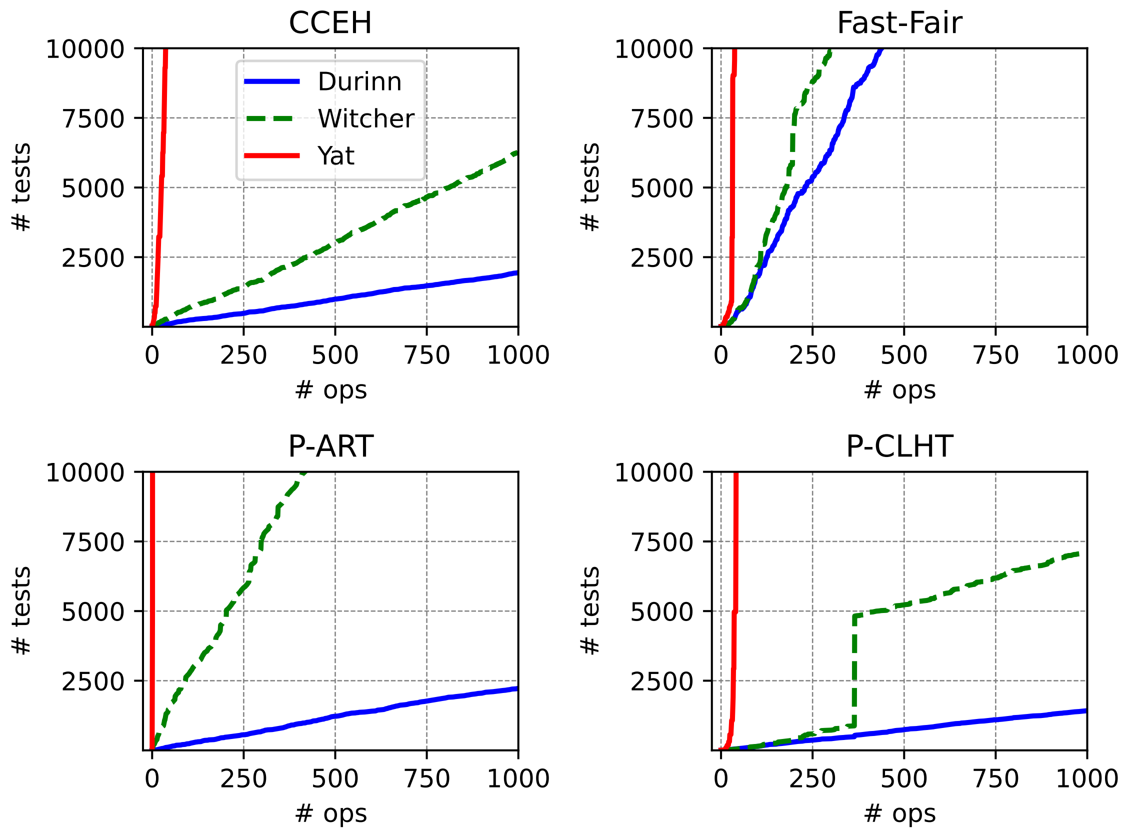


- Durinn only tests 35% and 82% of Total Stores
- Durinn did not miss true Linearization points

Comparison against Witcher

Bug Detection:

- Durinn reports 10 DL3 bugs that Witcher missed
- Durinn reduces the test space of thread interleaving



Test Space Reduction:

- Witcher performs several times more tests than Durinn
- Durinn only adversarially tests worst-case scenarios

Outline

- Introduction
- Durinn
- Evaluation
- ***Conclusion***

Conclusion

Durinn

- The first Durable Linearizability checker
- Three Durable Linearizability bug patterns
- Adversarial Crash State and Thread Interleaving Construction
- Likely-Linearization Point inference
- Detected 27 (15 new) bugs

Durinn: Adversarial Memory and Thread Interleaving for Detecting Durable Linearizability Bugs

Xinwei Fu*, Dongyoon Lee+, Changwoo Min*

* Virginia Tech

+ Stony Brook University