Crash Consistency in Block-Level Caching Systems: An Open CAS Case Study

Shaohua Duan

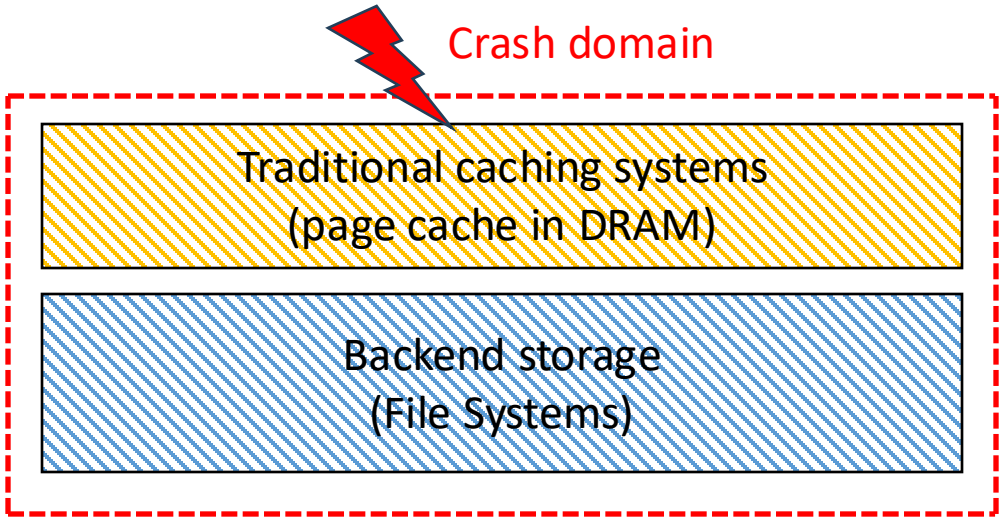
Youmin Chen

Outline

- Motivation
- Introduction of Open CAS
- Challenges for Testing Open CAS
- Workload-Oriented Test
- Test Results
- Beyond Open CAS (NVCache)
- Summary

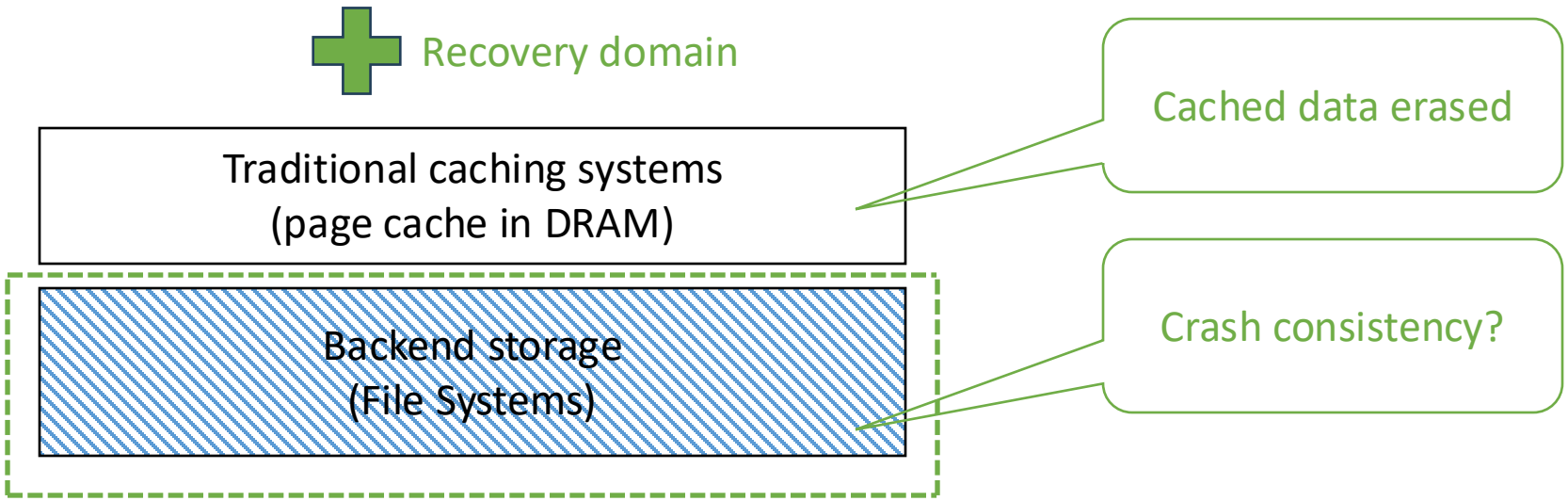
Why Crash Consistency Does Matter

- Traditional caching systems (e.g., page cache in DRAM)
  - ✓ Cached data is NOT available after a crash (simple crash model).



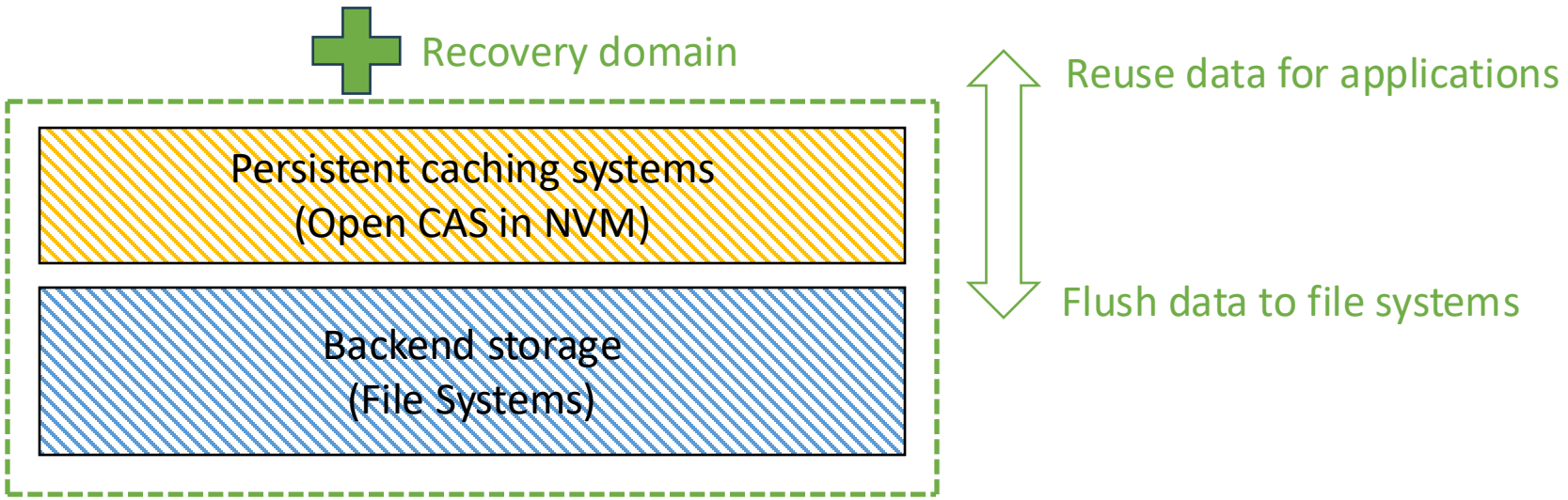
Why Crash Consistency Does Matter

- Traditional caching systems (e.g., page cache in DRAM)
  - ✓ Cached data is NOT available after a crash (simple crash model).
  - ✓ Crash consistency is ONLY for backend file systems.



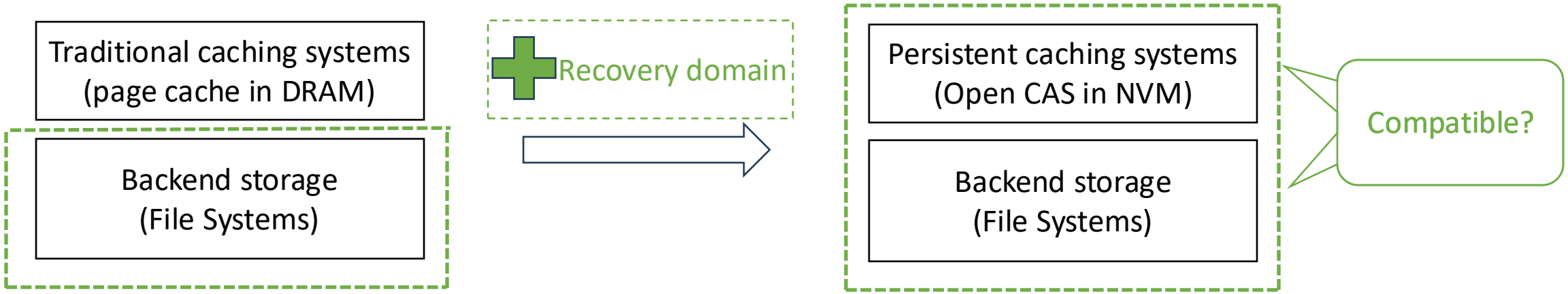
Why Crash Consistency Does Matter

- Persistent block-level caching systems (e.g., Open CAS in NVM)
  - ✓ Reuse the cached data for fast recovery.
  - ✓ Provide extra data durability for end-to-end file systems.



Why Crash Consistency Does Matter

- Potential compatibility issues
  - ✓ File system reliability features are designed on the stale crash model.
  - ✓ It remains unknown for the compatibility of post-recovery process between caching systems and backend file systems.

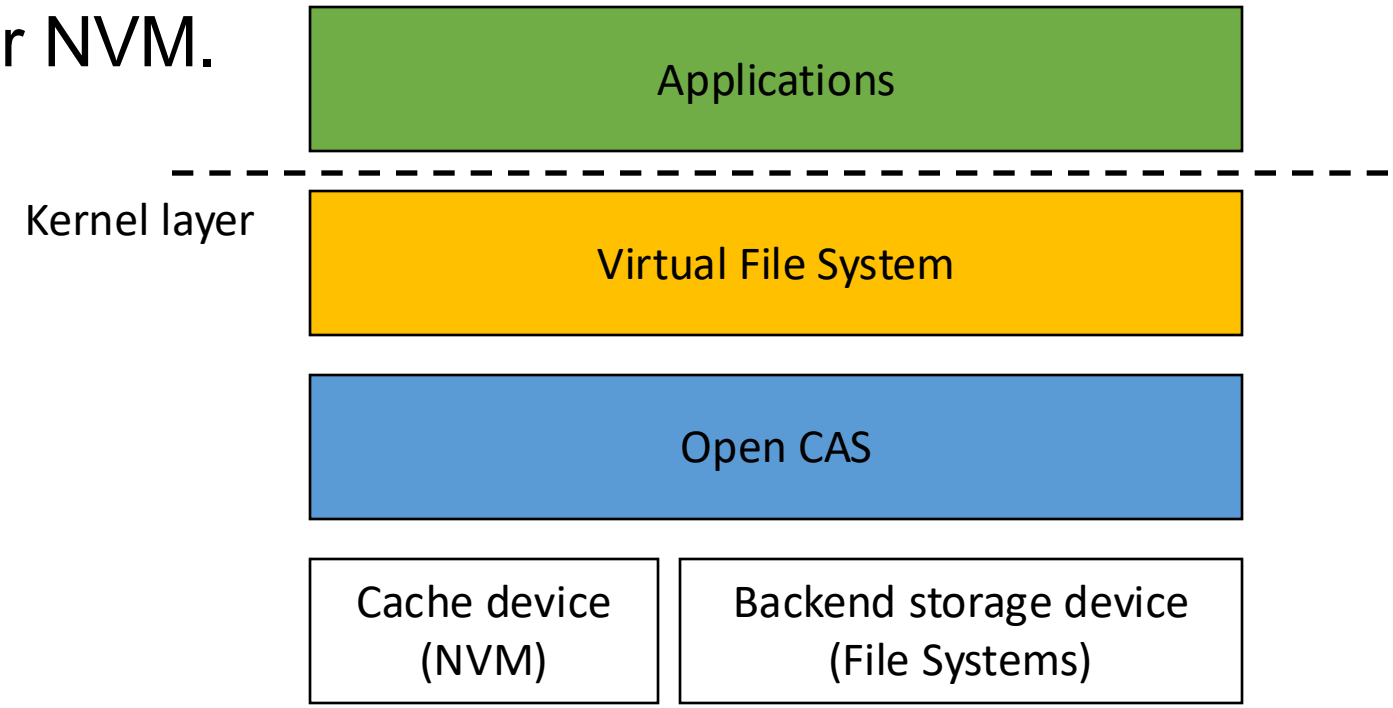


Why Crash Consistency Does Matter

Crash consistency in persistent caching layers and compatibility with file systems are important!

Introduction of Open CAS

- **Open CAS**: **O**pen **C**ache **A**cceleration **S**oftware (https://open-cas.github.io/) is a representative block-level caching system implemented as a kernel module for NVM.



Introduction of Open CAS

- **Open CAS**: **Open C**ache **A**cceleration **S**oftware (https://open-cas.github.io/) is a representative block-level caching system implemented as a kernel module for NVM.

	Cache Mode
1	Write-Through
2	Write-Back
3	Write-Around
4	Write-Invalidate
5	Write-Only

Open CAS Cache modes, cache states and cache operations

Introduction of Open CAS

- **Open CAS**: **Open C**ache **A**cceleration **S**oftware (https://open-cas.github.io/) is a representative block-level caching system implemented as a kernel module for NVM.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through					
2	Write-Back					
3	Write-Around					
4	Write-Invalidate					
5	Write-Only					

Open CAS Cache modes, cache states and cache operations

Introduction of Open CAS

- **Open CAS**: **Open C**ache **A**cceleration **S**oftware (https://open-cas.github.io/) is a representative block-level caching system implemented as a kernel module for NVM.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through	-	Update	Mapping, Insertion	Eviction, Invalidate	-
2	Write-Back					
3	Write-Around					
4	Write-Invalidate					
5	Write-Only					

Open CAS Cache modes, cache states and cache operations

Introduction of Open CAS

- **Open CAS**: **Open C**ache **A**cceleration **S**oftware (https://open-cas.github.io/) is a representative block-level caching system implemented as a kernel module for NVM.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through	-	Update	Mapping, Insertion	Eviction, Invalidate	-
2	Write-Back	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate
3	Write-Around	Invalidate	-	Mapping, Insertion	Eviction, Invalidate	-
4	Write-Invalidate	-	Update	Mapping, Insertion	Eviction, Invalidate	-
5	Write-Only	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate

(table is NOT complete)

Open CAS Cache modes, cache states and cache operations

Introduction of Open CAS

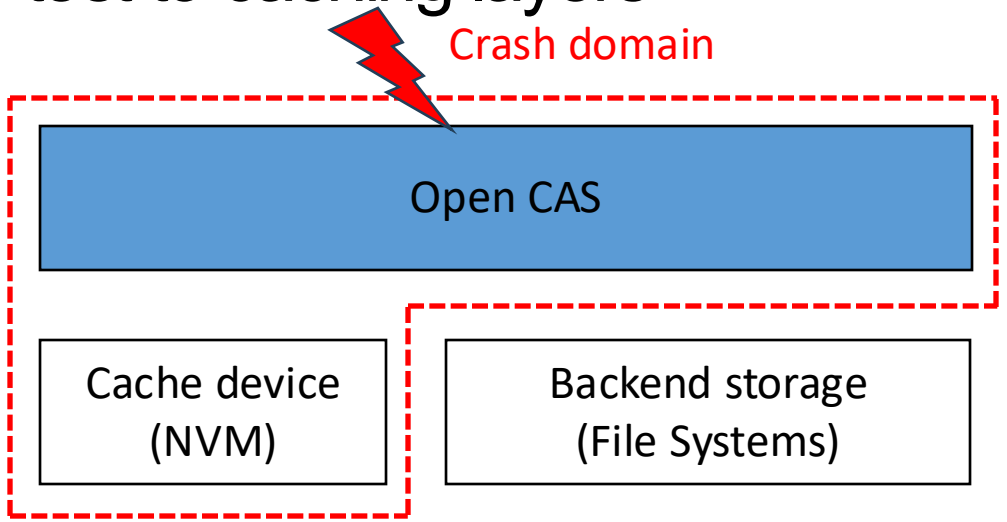
- Guarantee:
  - ✓ Crash consistency and data durability once returning an ACK to users

Introduction of Open CAS

- Guarantee:
  - ✓ Crash consistency and data durability once returning an ACK to users
- Reliability features:
  - ✓ Atomic metadata update
  - ✓ Data metadata I/O ordering
  - ✓ Flapped section
- Post-recover process:
  - ✓ Recover the cached data in caching layers for reuse

Crash Consistency Test to Open CAS

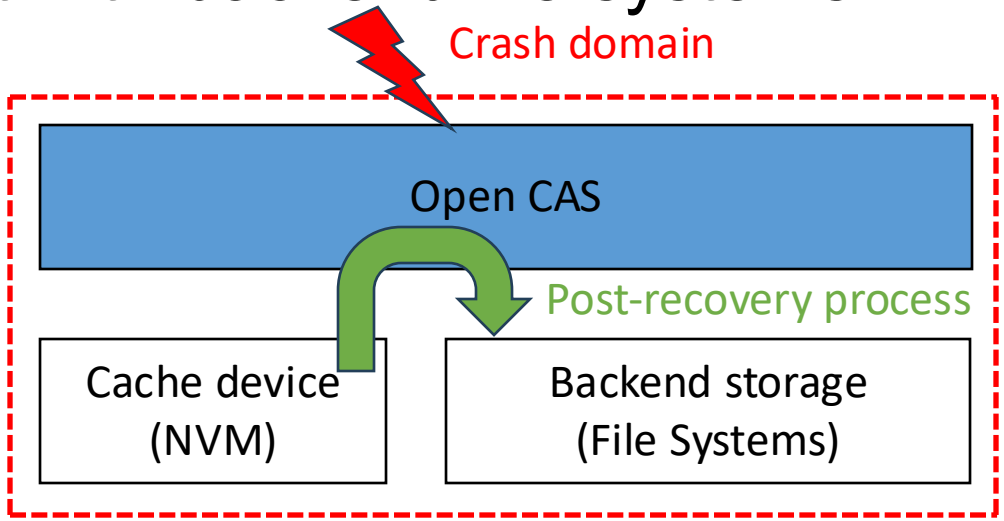
- Crash consistency test to caching layers



- Goals & Principles:
  - ✓ Explore crash consistency properties of caching layers at any time, including cache operations on the fly
  - ✓ Cover all cache operations, cache states, and cache modes

Crash Consistency Test to Open CAS

- Compatibility test with backend file systems



- Goals & Principles:
  - ✓ Perform the post-recovery process on both caching systems and backend file systems
  - ✓ Include diverse reliability features of file systems

Test Procedure

- **Workload**:
  - ✓ Leverage different workloads to cover all cache operations and modes
- **Crash Point**:
  - ✓ Perform crashes right after ACK and cache operations on the fly
- **File systems and reliability features** for the compatibility test:

File system	ext-2	ext-3	ext-4	btrfs	xfs
Reliability feature	sync	Data journal Metadata journal Write-back journal	Data journal Metadata journal Write-back journal	Copy-on-write	Copy-on-write

One challenge for testing Open CAS

- Implicit Cache Operation:
  - ✓ Implicitly be invoked, NO APIs for directly calling
  - ✓ Require heavy workload to be triggered through cache policies
  - ✓ Open CAS: *cache eviction, cache cleaning*

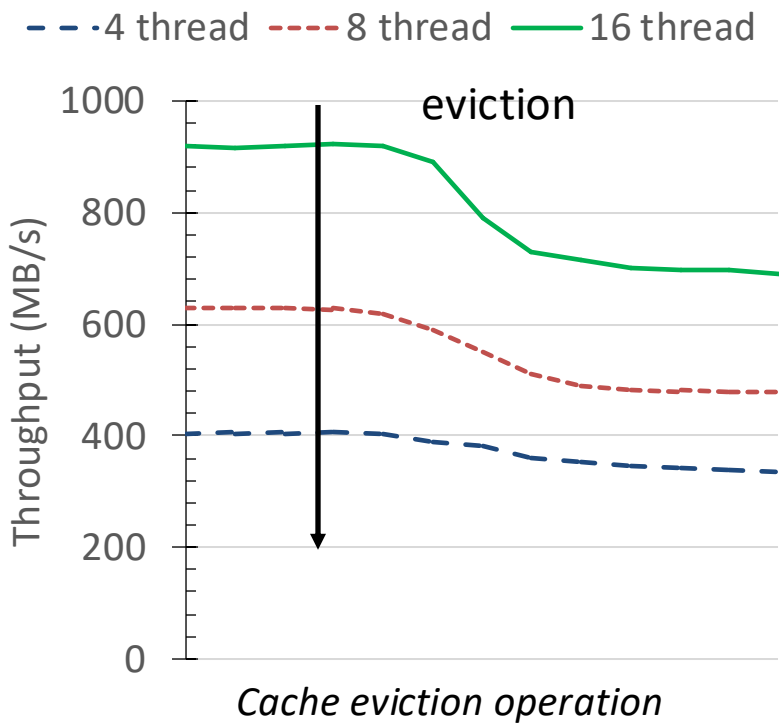
One challenge for testing Open CAS

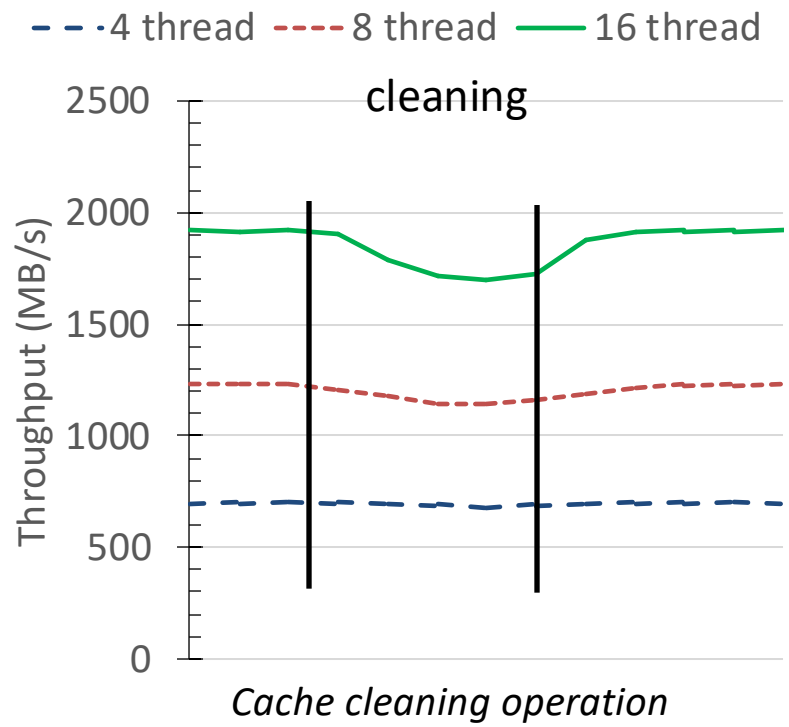
- Implicit Cache Operation:
  - ✓ Implicitly be invoked, NO APIs for directly calling
  - ✓ Require heavy workload to be triggered through cache policies
  - ✓ Open CAS: *cache eviction, cache cleaning*

How to precisely add crash points to implicit cache operations?

Implicit Cache Operation

- Performance degradation of implicit cache operation

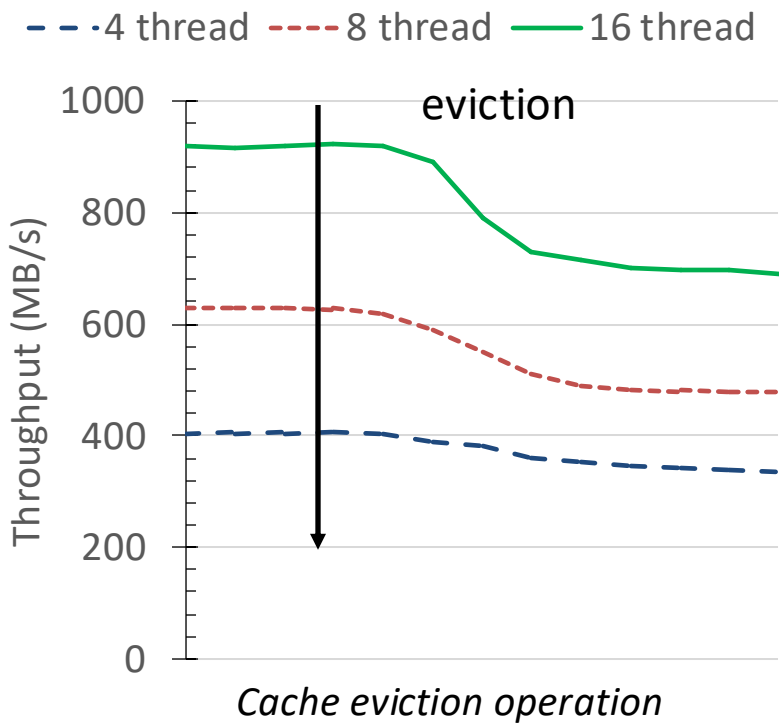


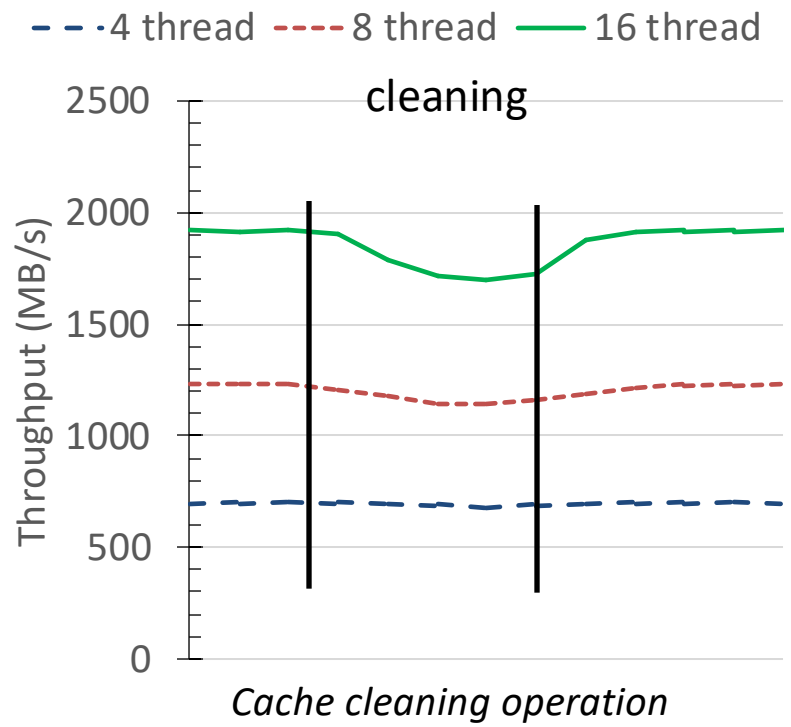


✓ **Intensive workload** (# threads)

Implicit Cache Operation

- Performance degradation of implicit cache operation

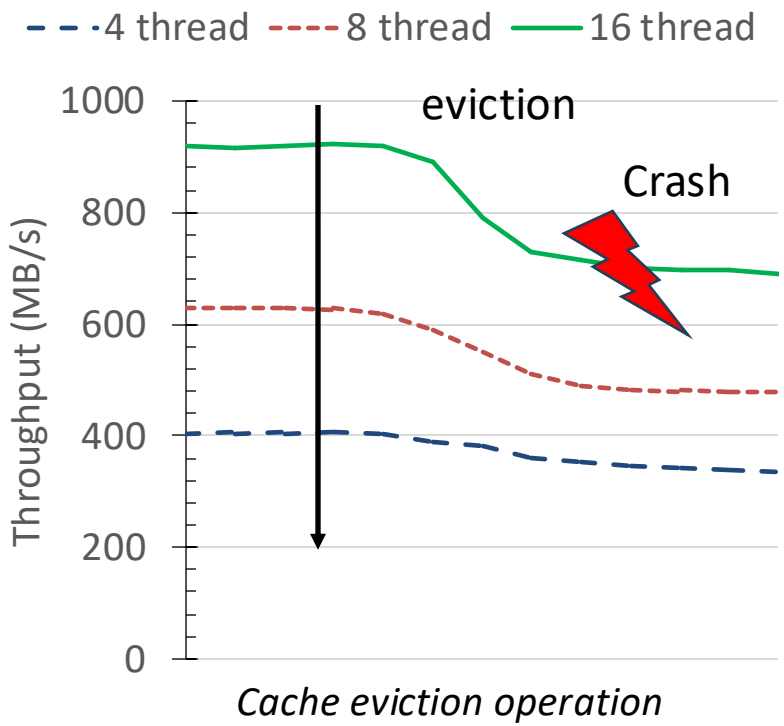


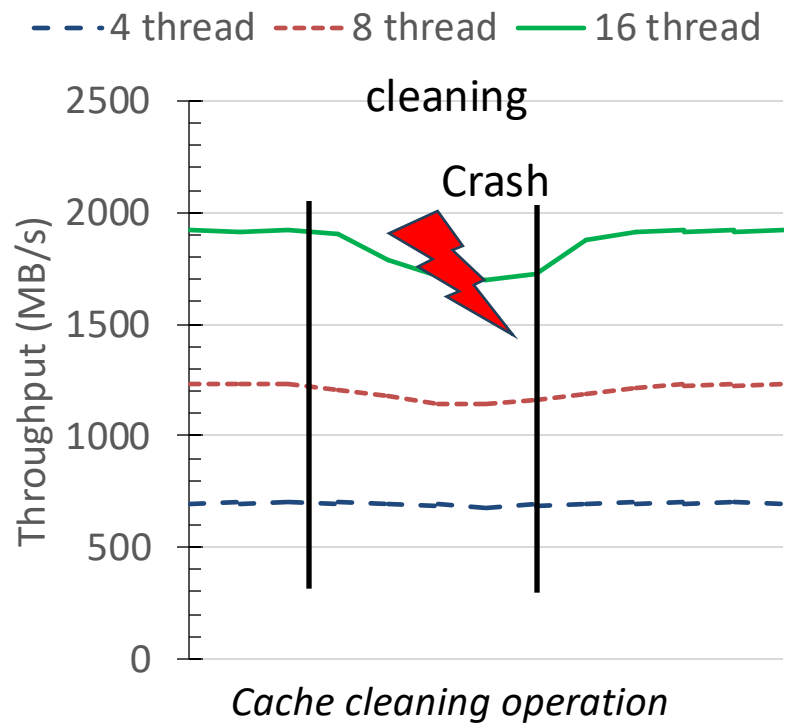


- ✓ **Intensive workload** (# threads)
- ✓ **Small cache line size**

Implicit Cache Operation

- Performance degradation of implicit cache operation

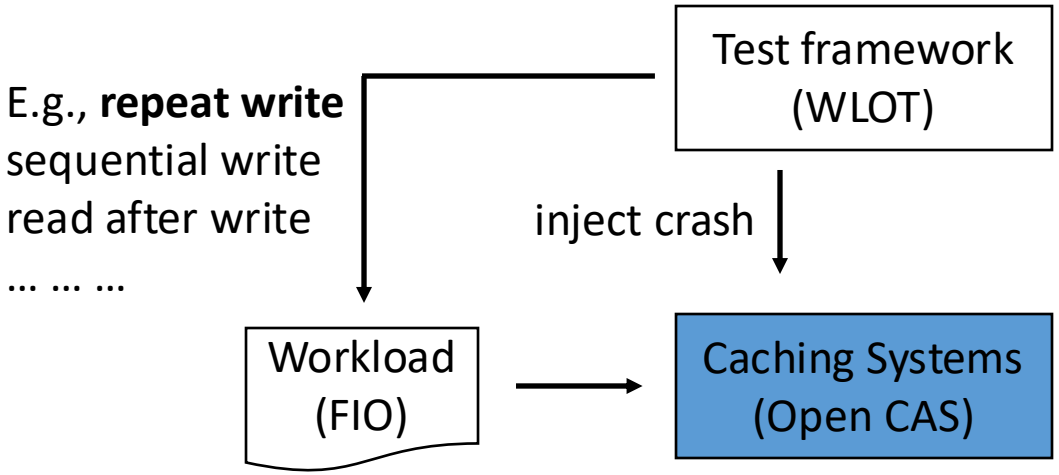




✓ **Intensive workload** (# threads) and **small cache line size** ease in identifying the state of cache operations and injecting crashes.

WorkLoad-Oriented Test (WLOT)

- Match cache operations with different workloads



Workflow of WLOT

Workload: Repeat write

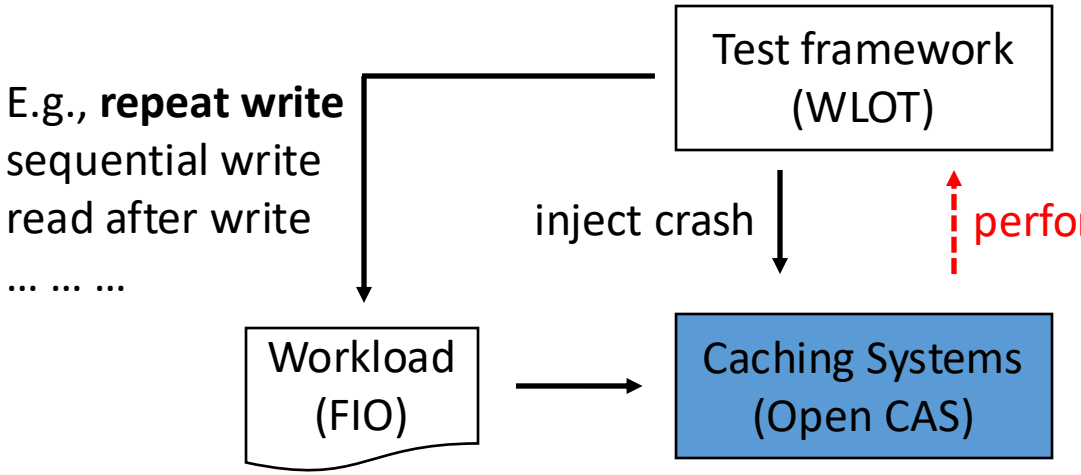
Mode: Write-back

State: Cache hit

Operation: Update

WorkLoad-Oriented Test (WLOT)

- Leverage performance degradation characteristics of implicit cache operation to inject synthetic crashes precisely.



Workflow of WLOT

Workload:	Repeat write	+ performance degradation
Mode:	Write-back	Write-back
State:	Cache hit	Cache cleaning
Operation:	Update	Cleaning, Invalidate

Crash Consistency Test Results

- Observation 1:

  Open CAS gracefully maintains crash consistency to caching layers in all cache operations and modes once it returns an acknowledgment to users.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through	-	Update	Mapping, Insertion	Eviction, Invalidate	-
2	Write-Back	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate
3	Write-Around	Invalidate	-	Mapping, Insertion	Eviction, Invalidate	-
4	Write-Invalidate	-	Update	Mapping, Insertion	Eviction, Invalidate	-
5	Write-Only	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate

(table is NOT complete)

Crash Consistency Test Results (cont.)

- Observation 2:

  Open CAS maintains crash consistency to caching layers if a crash happens during a cache miss and cache eviction.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through	-	Update	Mapping, Insertion	Eviction, Invalidate	-
2	Write-Back	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate
3	Write-Around	Invalidate	-	Mapping, Insertion	Eviction, Invalidate	-
4	Write-Invalidate	-	Update	Mapping, Insertion	Eviction, Invalidate	-
5	Write-Only	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate

(table is NOT complete)

Crash Consistency Test Results (cont.)

- Observation 3:

  Except for write-around mode, Open CAS returns bad data to users if a crash happens during a write cache hit.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through	-	Update	Mapping, Insertion	Eviction, Invalidate	-
2	Write-Back	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate
3	Write-Around	Invalidate	-	Mapping, Insertion	Eviction, Invalidate	-
4	Write-Invalidate	-	Update	Mapping, Insertion	Eviction, Invalidate	-
5	Write-Only	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate

(table is NOT complete)

Crash Consistency Test Results (cont.)

- Observation 4:

  Open CAS returns bad data to users if a crash happens during a write request to the cached data in write-around mode.

	Cache Mode	Read cache hit	Write cache hit	Read/Write cache miss	Cache eviction	Cache cleaning
1	Write-Through	-	Update	Mapping, Insertion	Eviction, Invalidate	-
2	Write-Back	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate
3	Write-Around	Invalidate	-	Mapping, Insertion	Eviction, Invalidate	-
4	Write-Invalidate	-	Update	Mapping, Insertion	Eviction, Invalidate	-
5	Write-Only	-	Update	Mapping, Insertion	Eviction, Invalidate	Cleaning, Invalidate

(table is NOT complete)

Compatibility Test Results

- Observation 1:

  Performing the post-recovery process and flushing recovered data into the file systems maintains durability with a Slow Recovery (SR).

File system	Configurations	Cache mode	Result
ext-2	-, sync	Write-Back	SR
ext-3	-	Write-Back	SR
ext-3	writeback, ordered	Write-Back	FE
ext-3	journal	Write-Back	FE, LE
ext-4	nodelalloc, writeback, ordered	Write-Back	FE
ext-4	journal	Write-Back	FE, LE
btrfs	-	Write-Back	FE, LS
xfs	-, wsync	Write-Back	FE, LS

Compatibility Test Results (cont.)

- Observation 2:

  Performing the post-recovery process and Flushing the recovered data to the file systems incur I/O Error (FE).

File system	Configurations	Cache mode	Result
ext-2	-, sync	Write-Back	SR
ext-3	-	Write-Back	SR
ext-3	writeback, ordered	Write-Back	FE
ext-3	journal	Write-Back	FE, LE
ext-4	nodelalloc, writeback, ordered	Write-Back	FE
ext-4	journal	Write-Back	FE, LE
btrfs	-	Write-Back	FE, LS
xfs	-, wsync	Write-Back	FE, LS

Compatibility Test Results (cont.)

- Observation 3:

  Performing the post-recovery process and flushing the recovered data to the file systems incur durability Loss with an Error (LE).

File system	Configurations	Cache mode	Result
ext-2	-, sync	Write-Back	SR
ext-3	-	Write-Back	SR
ext-3	writeback, ordered	Write-Back	FE
ext-3	journal	Write-Back	FE, LE
ext-4	nodelalloc, writeback, ordered	Write-Back	FE
ext-4	journal	Write-Back	FE, LE
btrfs	-	Write-Back	FE, LS
xfs	-, wsync	Write-Back	FE, LS

Compatibility Test Results (cont.)

- Observation 4:

  Performing the post-recovery process and flushing the recovered to the file systems incur durability Loss Silently (LS).

File system	Configurations	Cache mode	Result
ext-2	-, sync	Write-Back	SR
ext-3	-	Write-Back	SR
ext-3	writeback, ordered	Write-Back	FE
ext-3	journal	Write-Back	FE, LE
ext-4	nodelalloc, writeback, ordered	Write-Back	FE
ext-4	journal	Write-Back	FE, LE
btrfs	-	Write-Back	FE, LS
xfs	-, wsync	Write-Back	FE, LS

Beyond Open CAS

- **NVCache** [1]: A simpler caching system

✓ A write cache to improve the write performance of applications in NVM.

	Cache Mode	Write cache hit	Write cache miss	Cache cleaning
1	Write-Back	Cache write	Cache write	Cleanup

✓ Guarantees synchronous durability using log structures for persisting cache line and metadata in the caching layer.

[1] *Remi Dulong, etc: A plug-and-play nvmm-based i/o booster for legacy systems. In Proceedings of the 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 21), page 186–198. IEEE, September 2021.*

Beyond Open CAS

- **NVCache** test results:

Cache Mode	Workload	Crash Point	Cache Operation	Result 1	Result 2
Write-Back	Sequential Write	Write cache miss	Cache write	Recovery	Consistency
		Cache cleaning	Cleanup	-	Bad Data
	Repeat Write	Write cache hit	Cache write	Recovery	Consistency

Result 1: Results of crashes right after acknowledgments
Result 2: Results of crashes when operations on the fly

Results Summary

- Open CAS and NVCache cannot maintain crash consistency in all crash scenarios. The corrupted data may return to users and/or be flushed to the backend file system.

- Open CAS fails to be compatible with the diverse reliability features of state-of-the-art file systems.

Thanks & Questions