FAST'22
FEBRUARY 22-24, 2022
SANTA CLARA, CA

exF2FS: Transaction Support in Log-Structured Filesystem

Joontaek Oh, Sion Ji, Yongjin Kim, Youjip Won

Department of Electrical Engineering

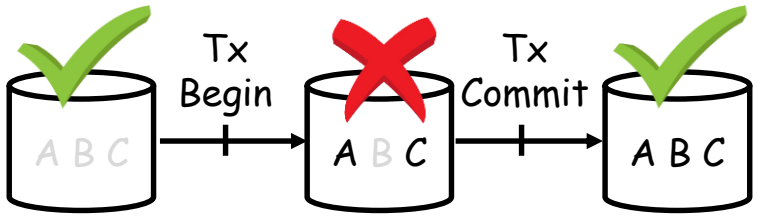
KAIST

What is a transaction?

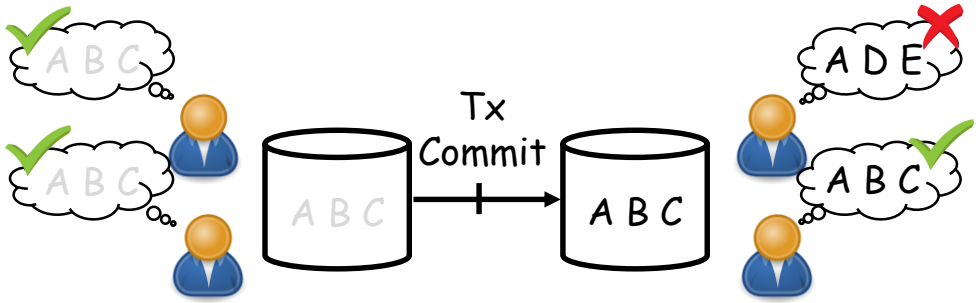
Transaction: a set of updates satisfying ACID properties

```
Tx_BEGIN;
  write A;
  write B;
  write C;
Tx_COMMIT;
```

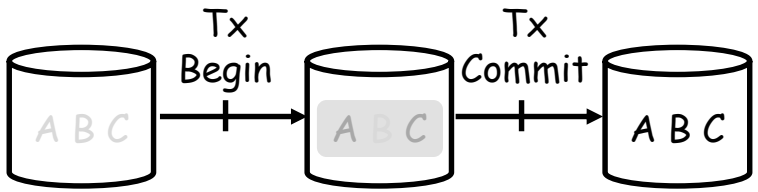
Atomicity:



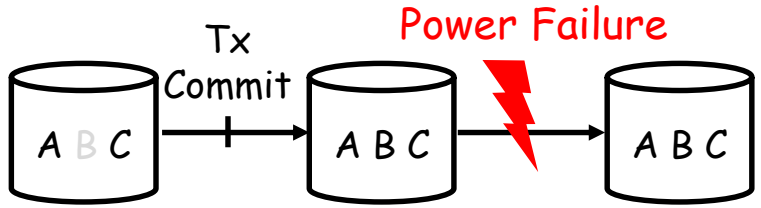
Consistency:



Isolation:

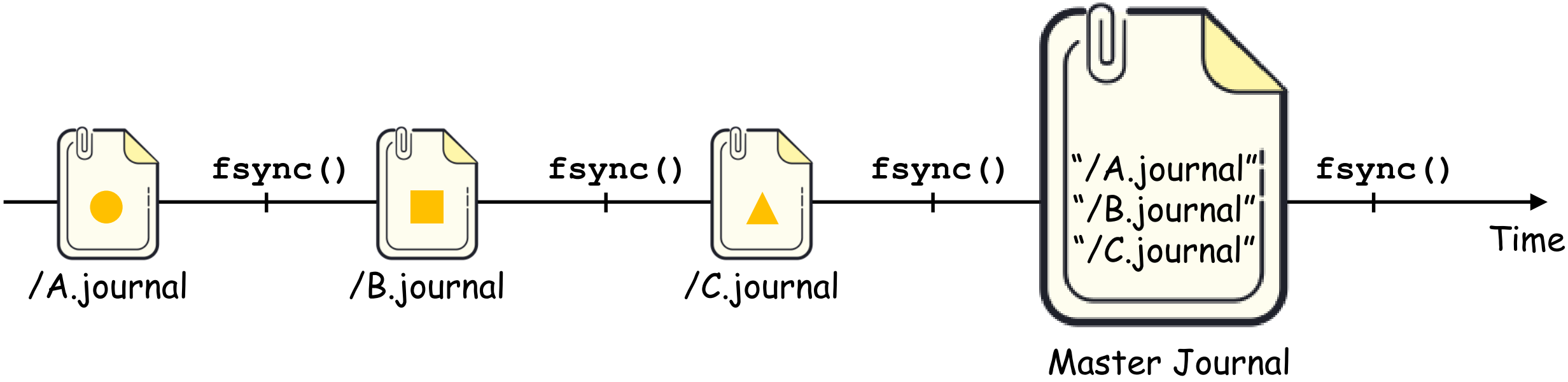


Durability:

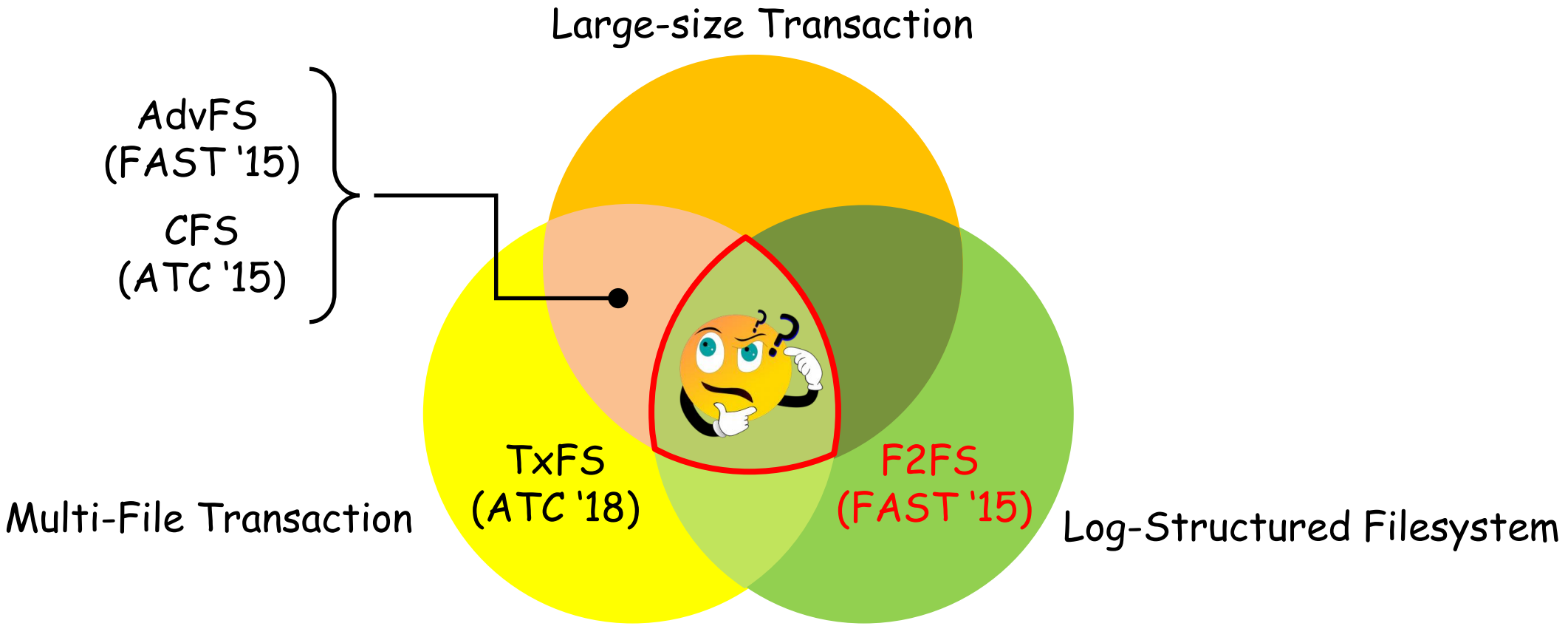


Transaction in the absence of filesystem-level transaction support

```
Tx_BEGIN;
  write(File "/A", ●);
  write(File "/B", ■);
  write(File "/C", ▲);
Tx_COMMIT;
```

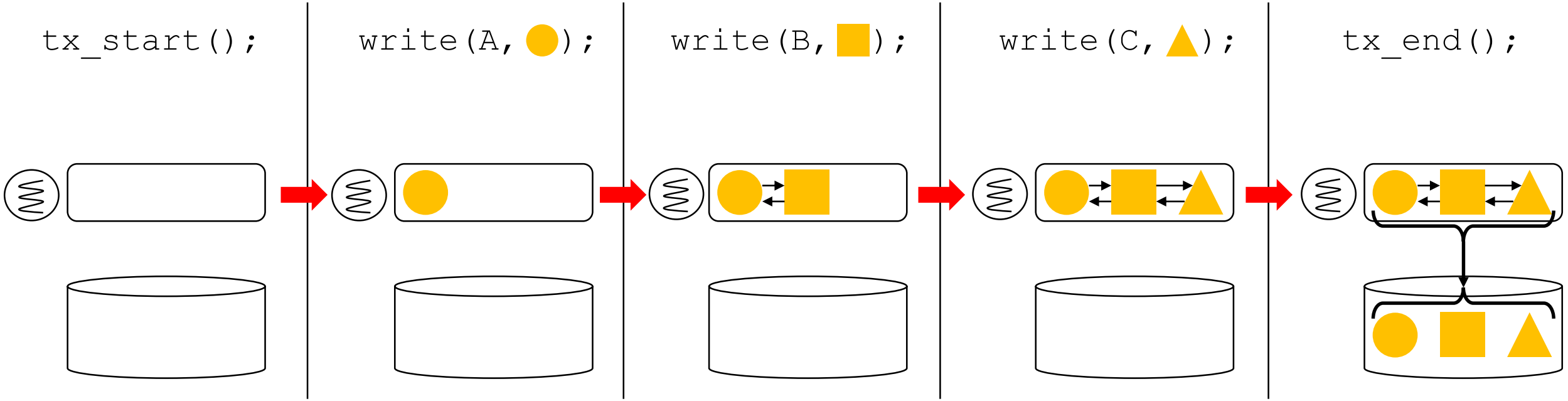


Large-sized Multi-File Transaction in Log-Structured Filesystem



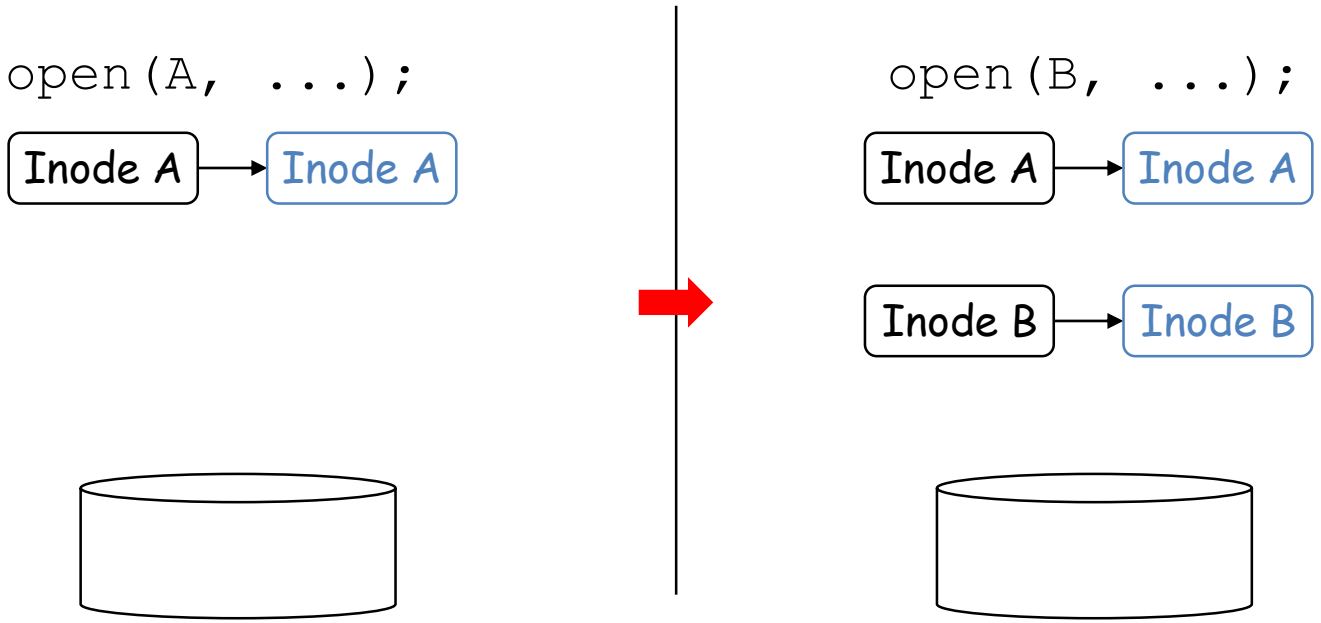
Multi-file transaction (TxFS and CFS)

```
tx_start();
write(File A, ●);
write(File B, ■);
write(File C, ▲);
tx_end();
```



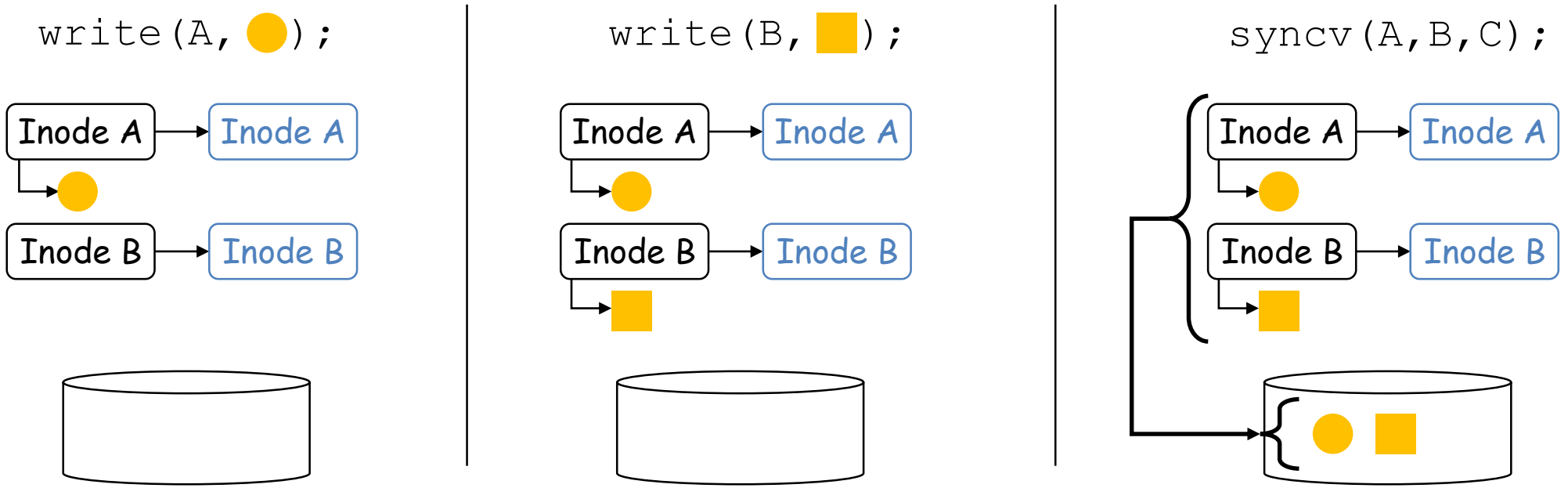
Multi-file transaction (AdvFS) (1)

```
open(File A, O_ATOMIC);
open(File B, O_ATOMIC);
write(File A, );
write(File B, );
syncv({File A, File B});
```



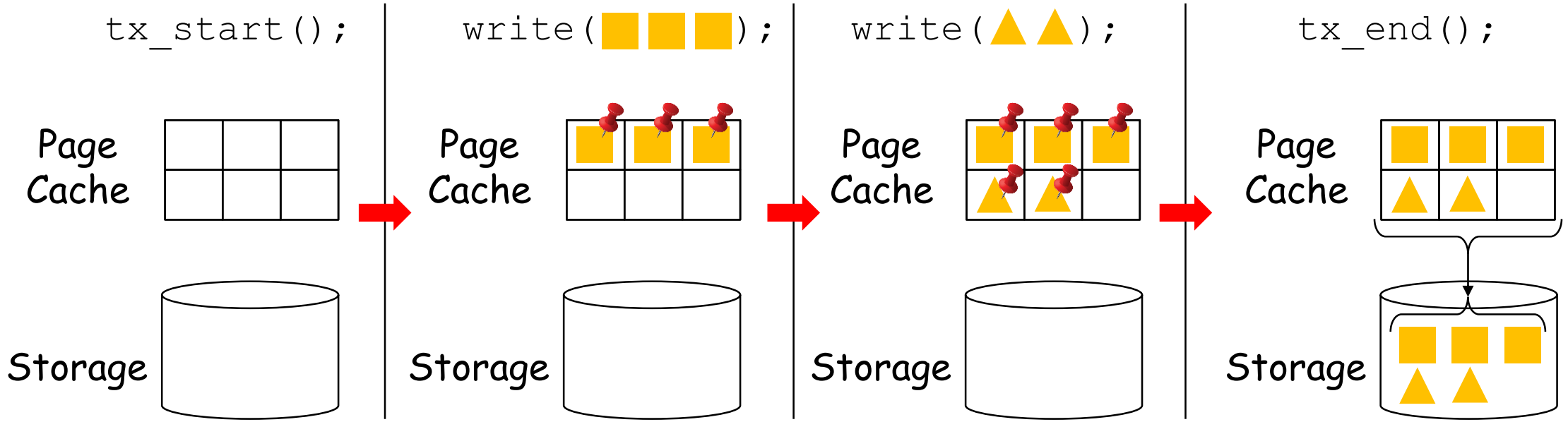
Multi-file transaction (AdvFS) (2)

```
open(File A, O_ATOMIC);
open(File B, O_ATOMIC);
write(File A, ●);
write(File B, ■);
syncv({File A, File B});
```



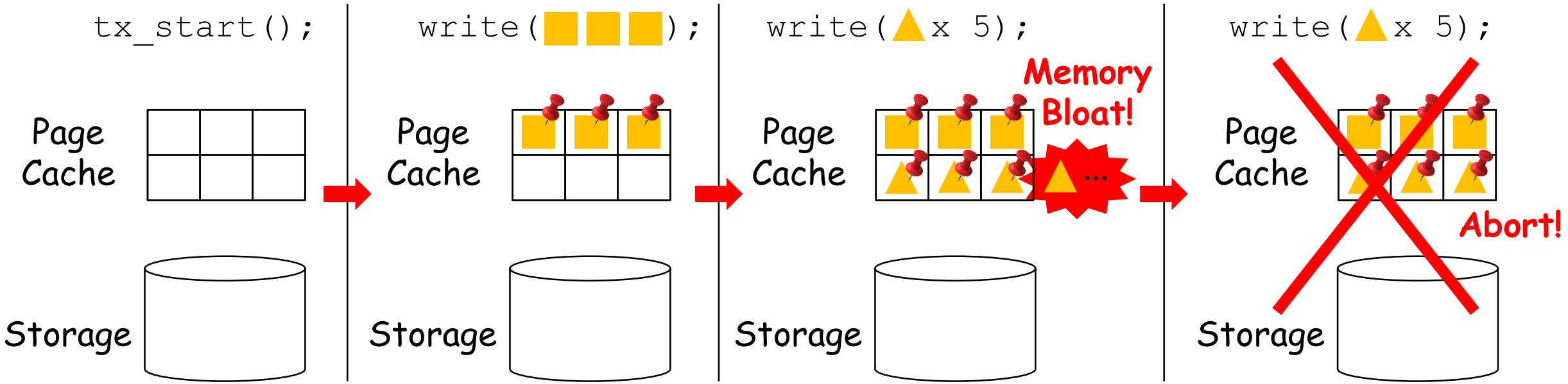
Transaction (F2FS and TxFS)

```
tx_start();
write(fd, ■ ■ ■ );
write(fd, ▲ ▲ );
tx_end();
```



Large Transaction (TxFS and F2FS)

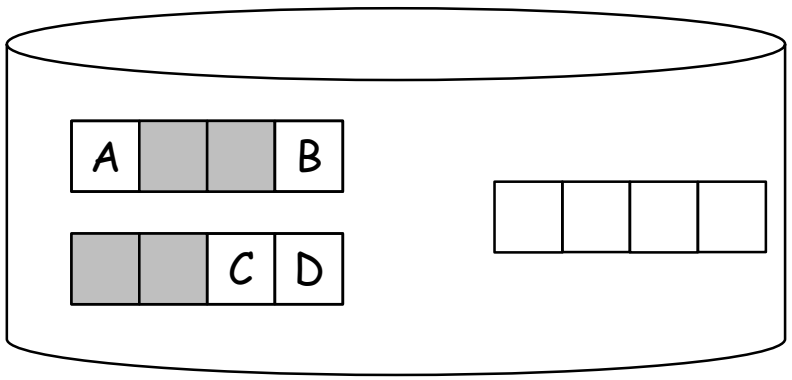
```
tx_start();
write(fd, ■ ■ ■ );
write(fd, ▲ ▲ ▲ ▲ ▲ );
tx_end();
```



Garbage Collection

File Mapping

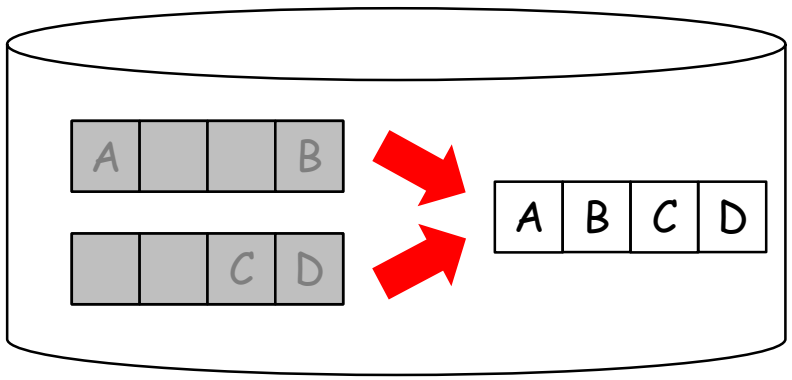
A: 1	C: 7
B: 4	D: 8



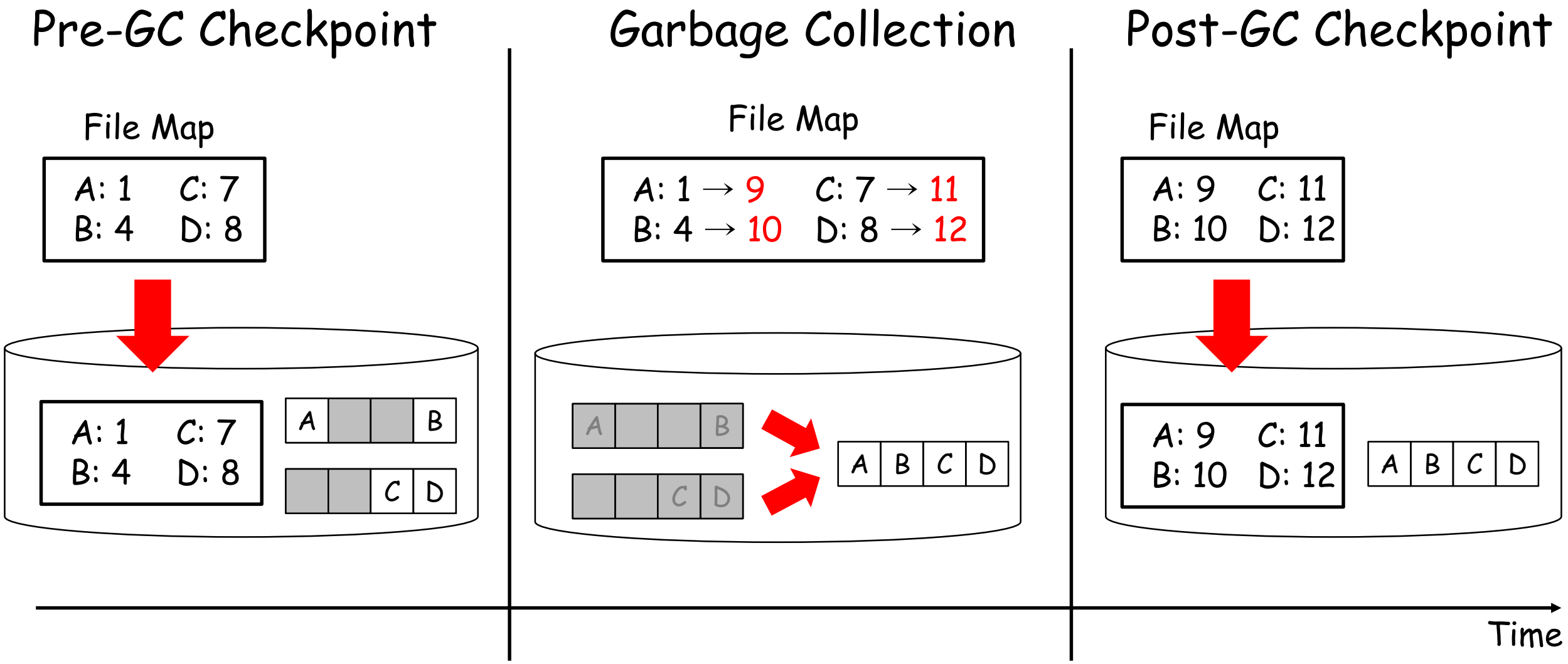
Garbage Collection

File Mapping

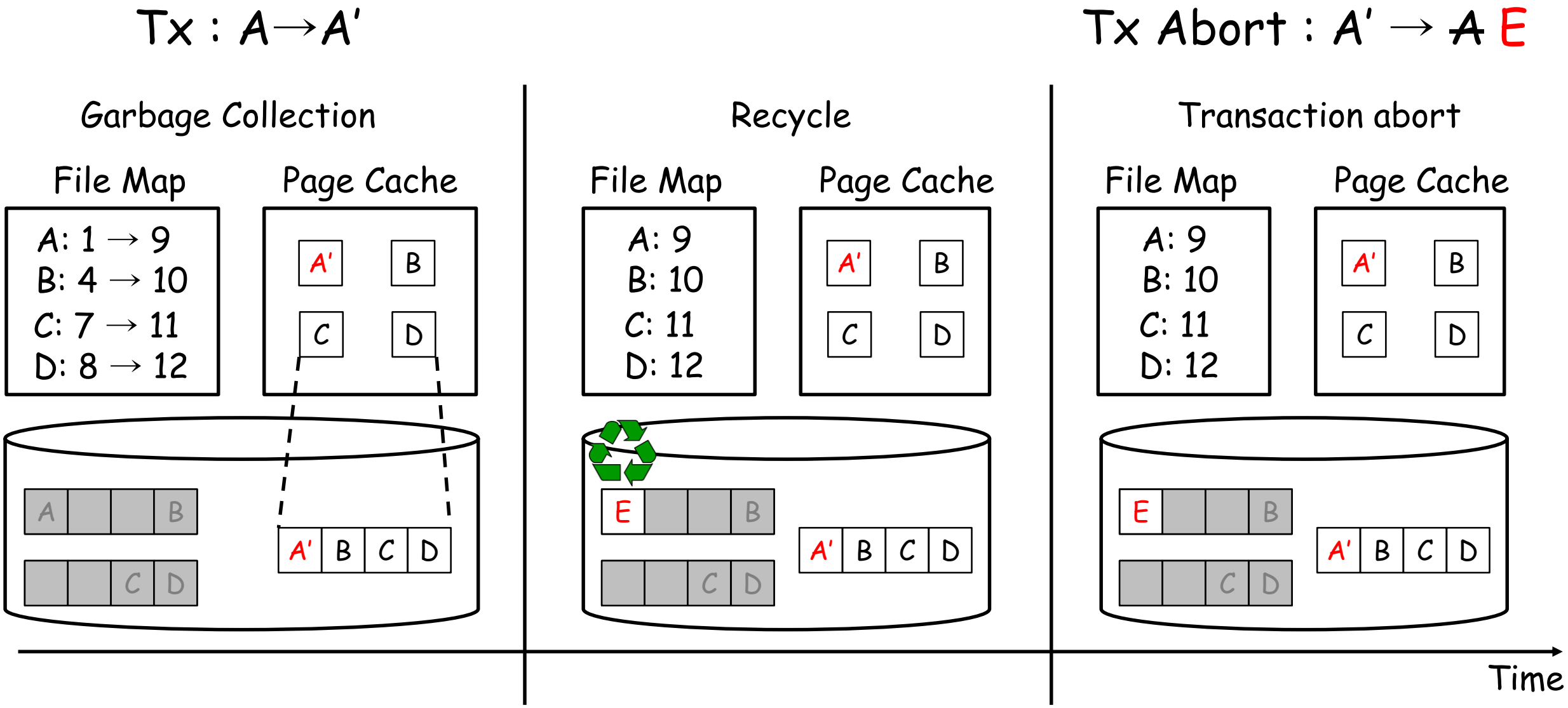
A: 9	C: 11
B: 10	D: 12



Garbage Collection in Log-Structured Filesystem



Transaction and Garbage Collection

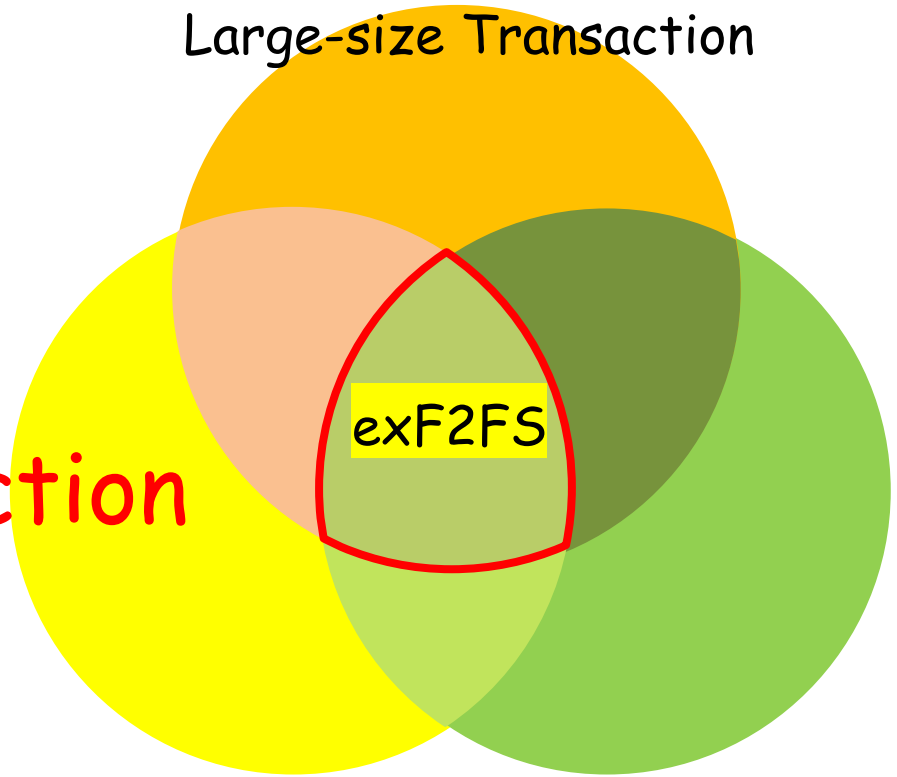


exF2FS

New Transactional Log-Structured Filesystem

Stealing

Large-size Transaction



Membership-Oriented Transaction

Multi-File Transaction

Shadow Garbage Collection

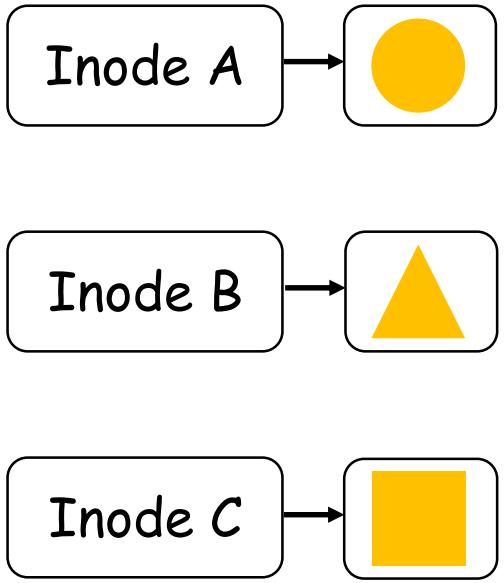
Log-Structured Filesystem

Membership-Oriented Transaction

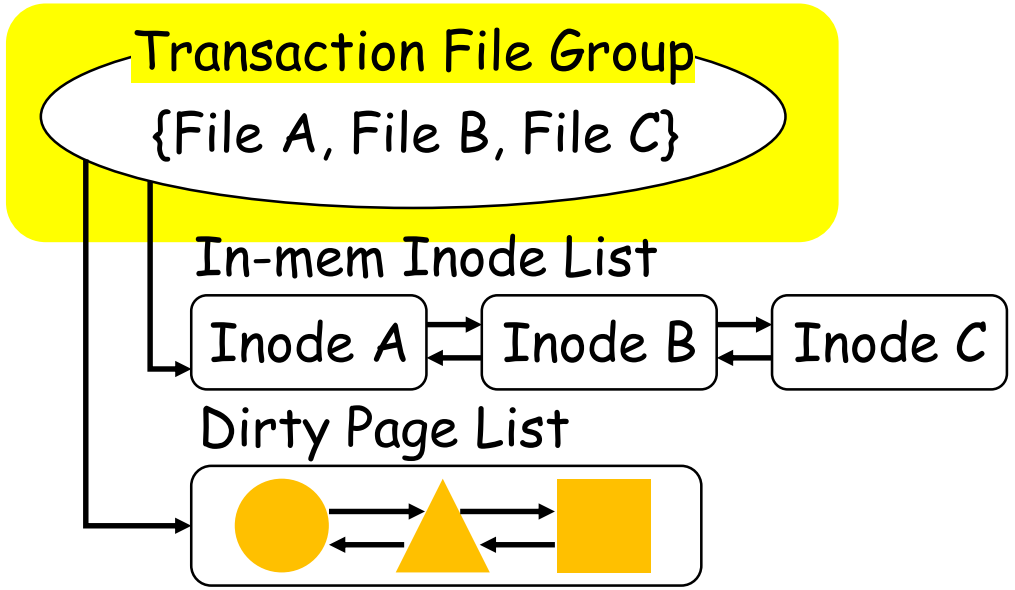
Membership-Oriented Transaction

```
Tx_BEGIN;
  write(File A, ●);
  write(File B, ■);
  write(File C, ▲);
Tx_COMMIT;
```

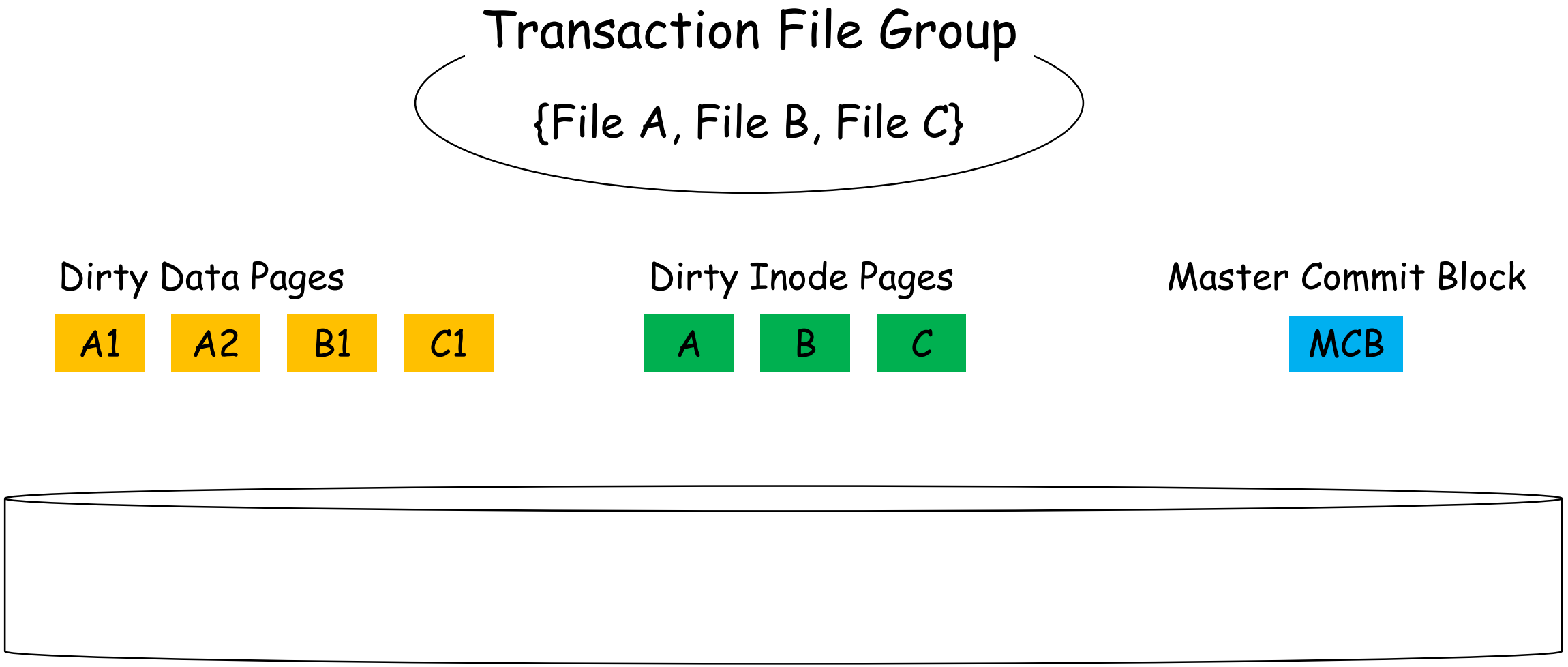
Transaction Model in F2FS:



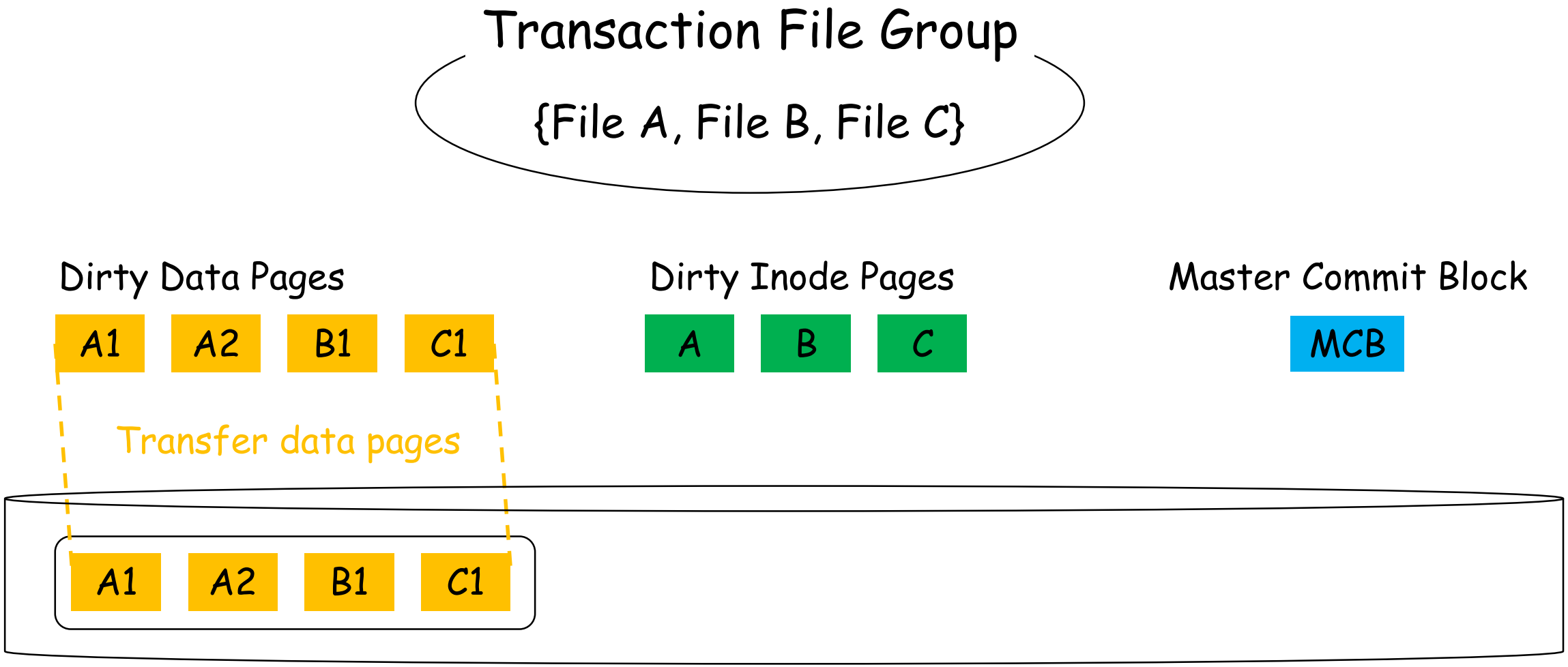
Membership-Oriented Transaction



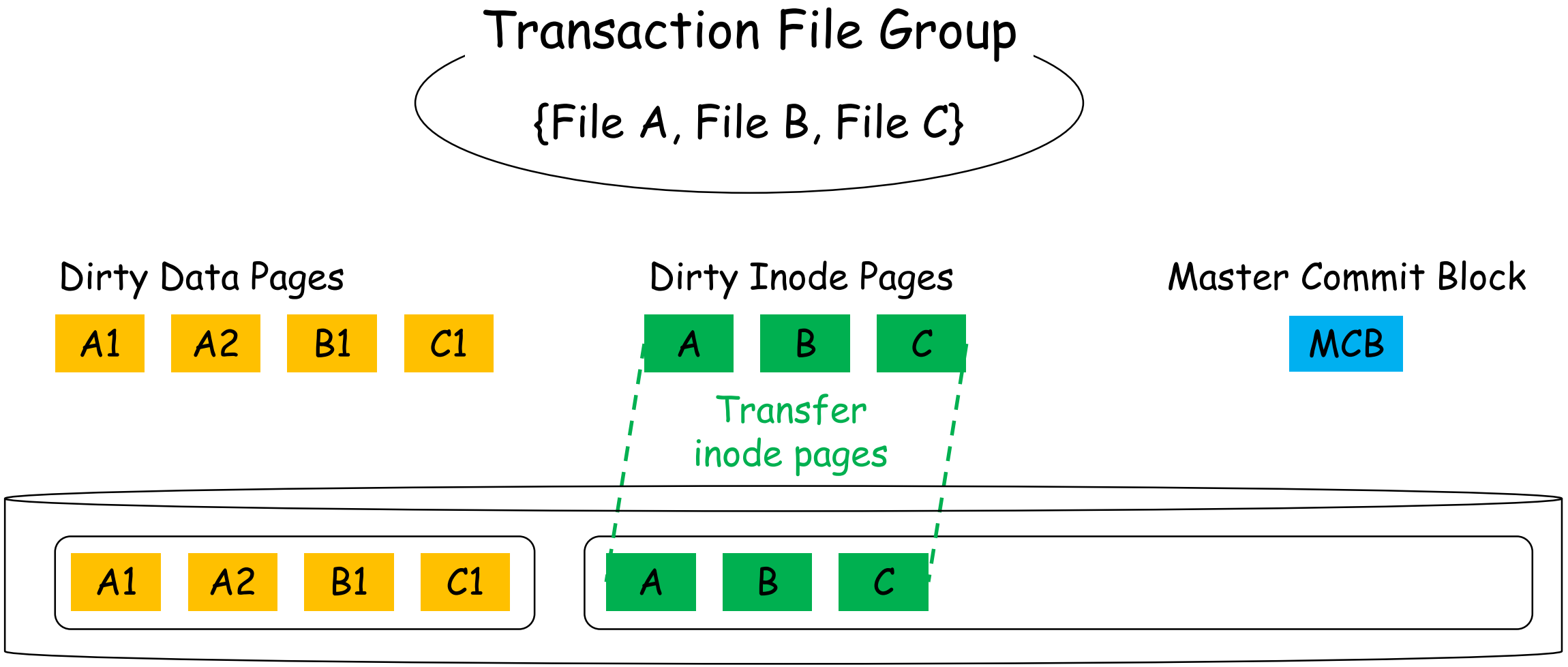
Commit in Membership-Oriented Transaction



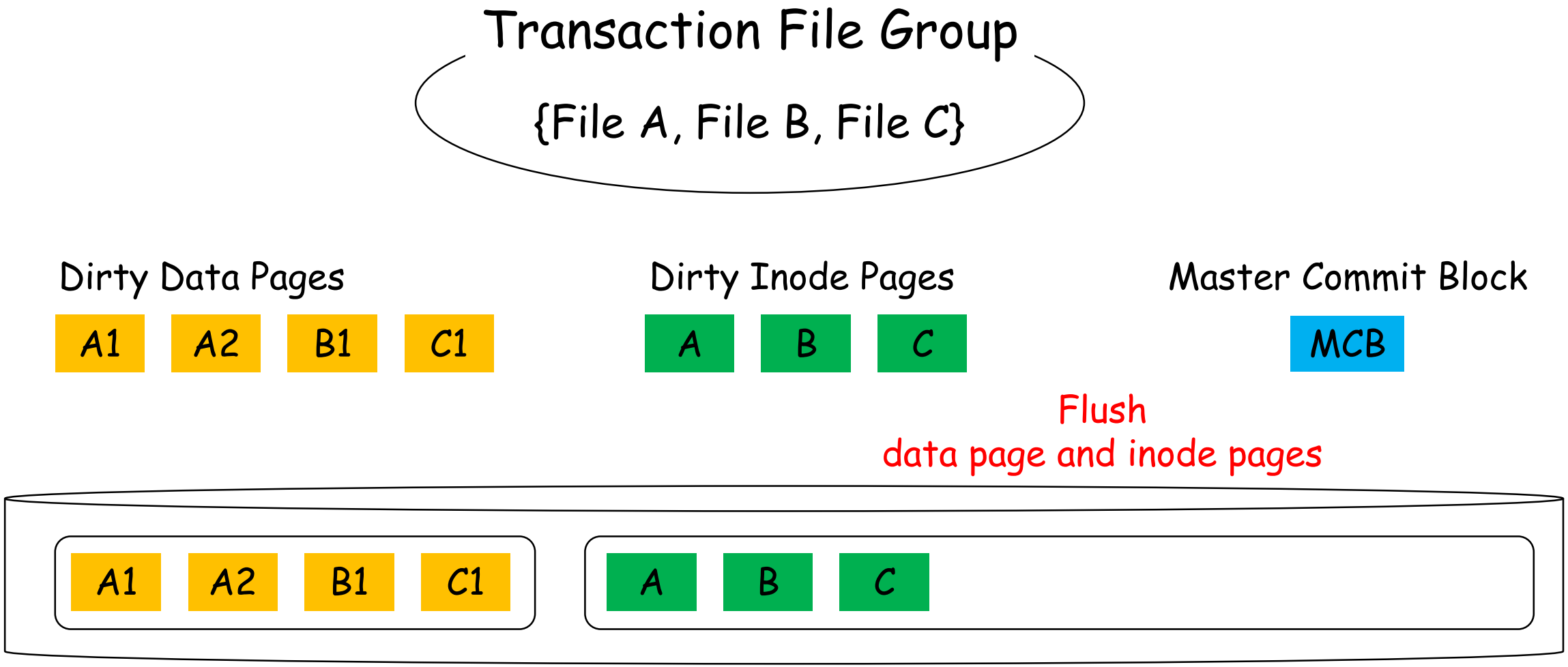
Commit in Membership-Oriented Transaction



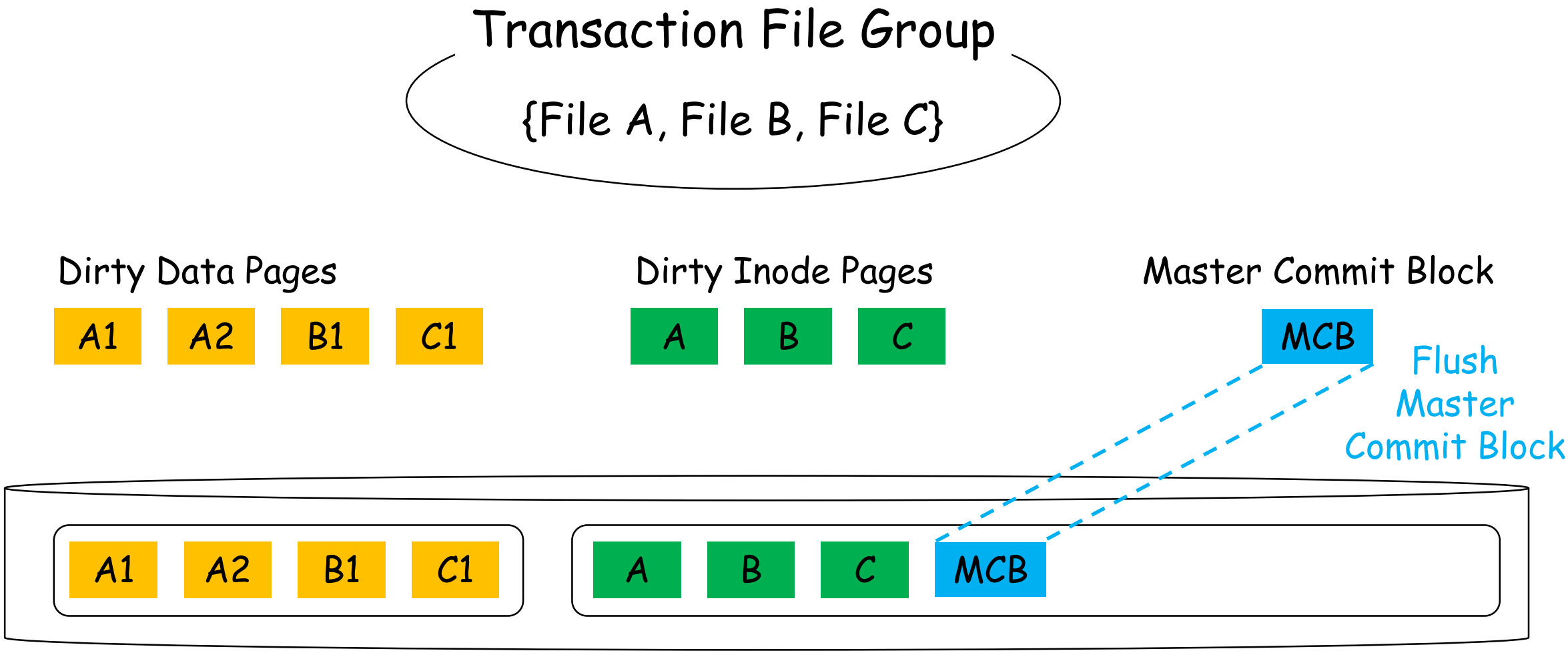
Commit in Membership-Oriented Transaction



Commit in Membership-Oriented Transaction



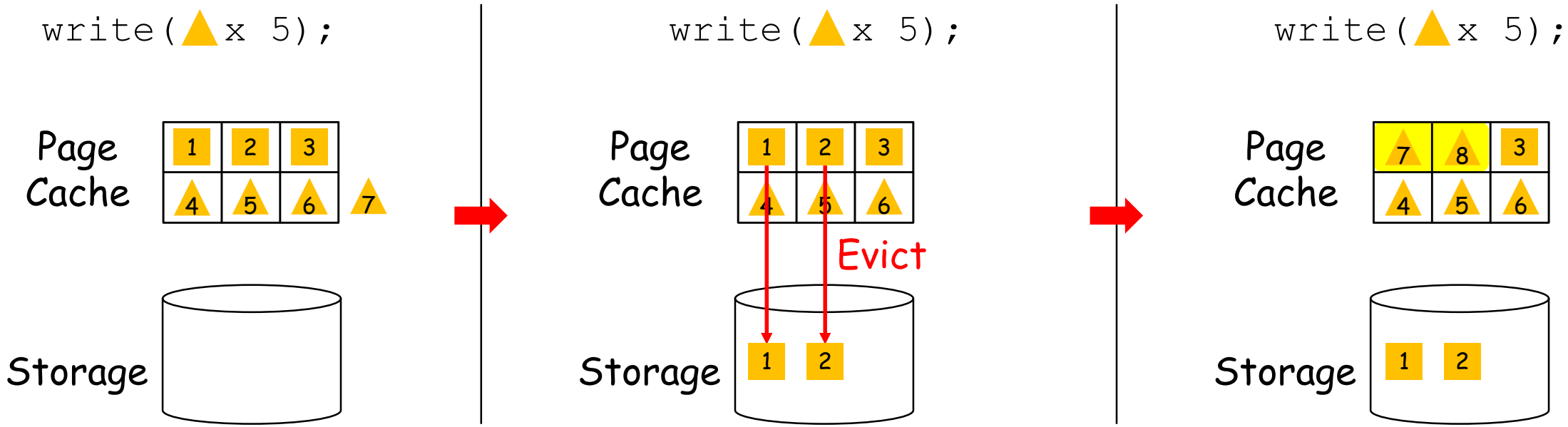
Commit in Membership-Oriented Transaction



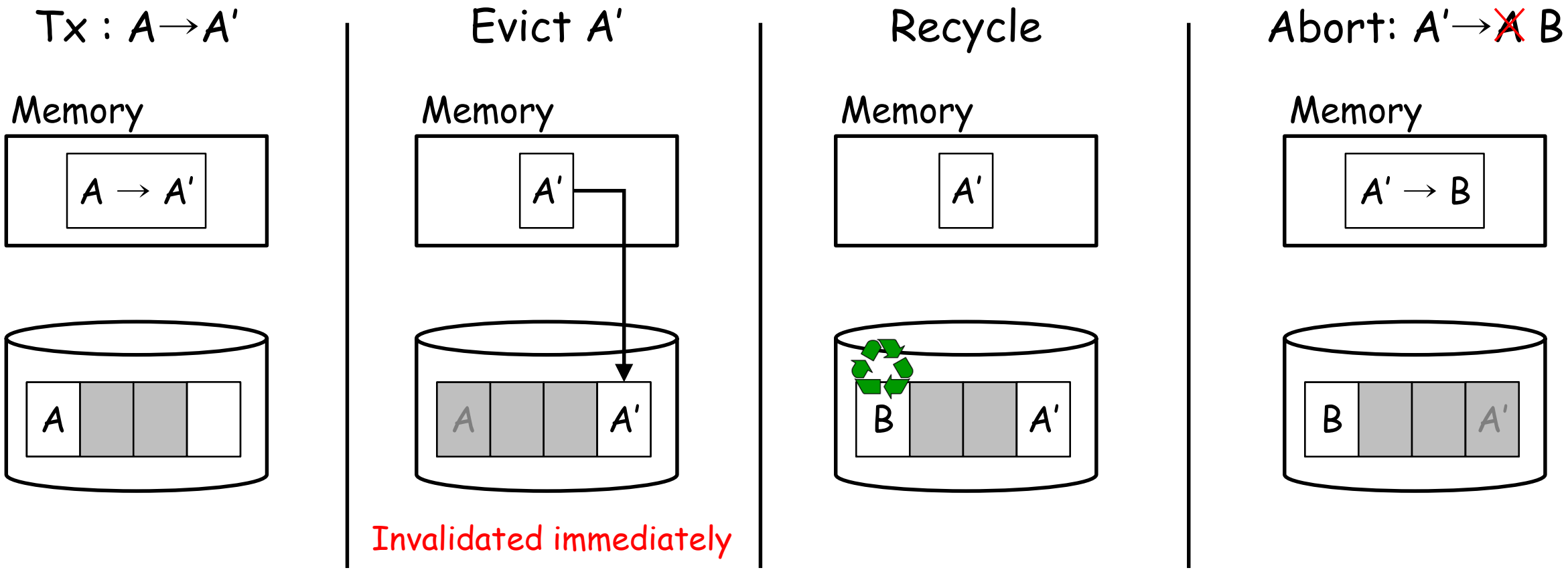
Stealing

Stealing–Enabled Transaction

```
tx_start();
write(fd, 1 2 3 );
write(fd, 4 5 6 7 8 );
tx_end();
```

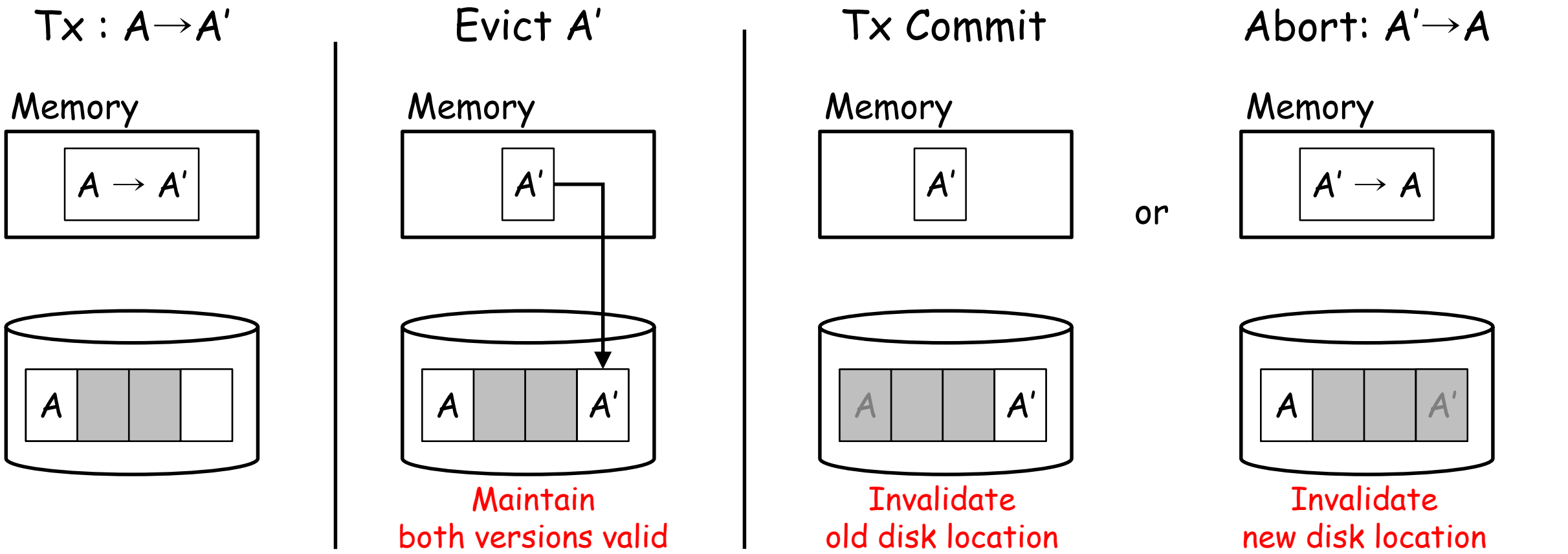


Immediate invalidation of the old disk location

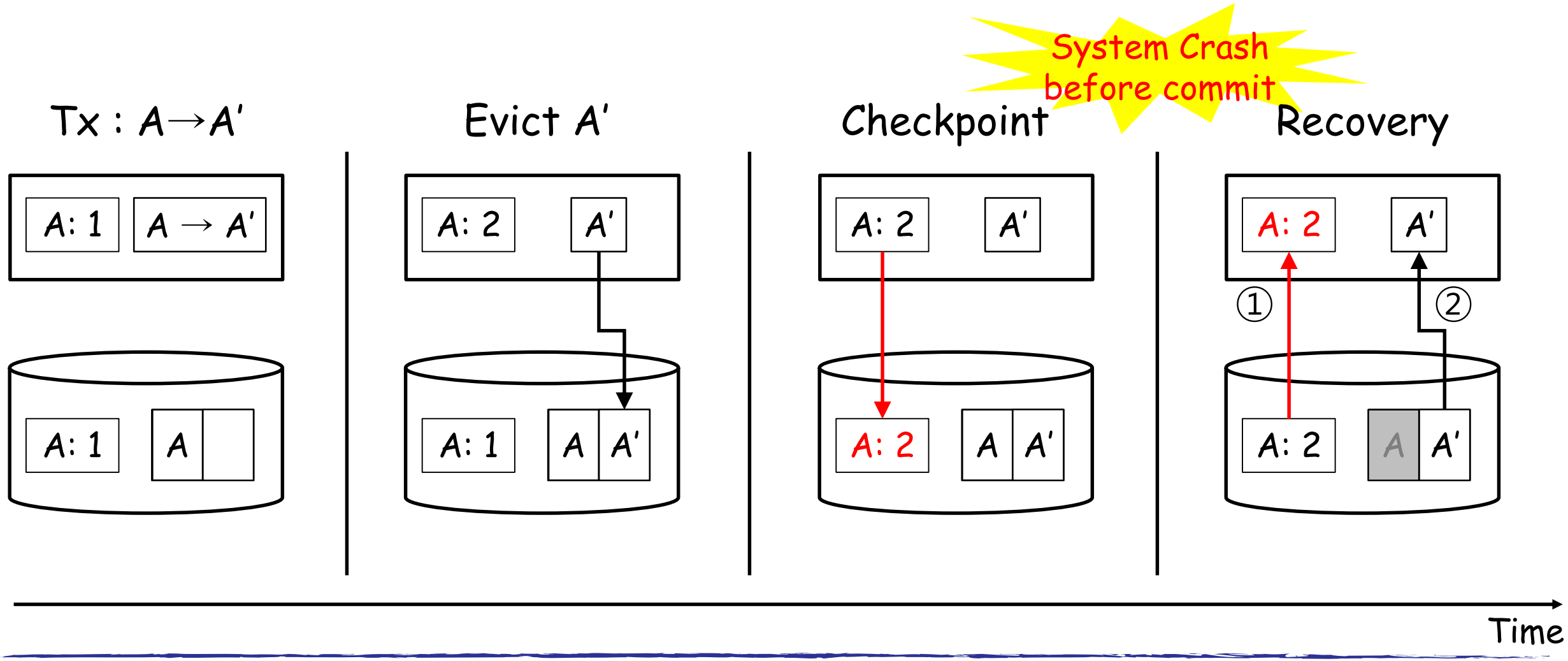


Delayed Invalidation

Delay the invalidation of evicted pages until the commit of the transaction.

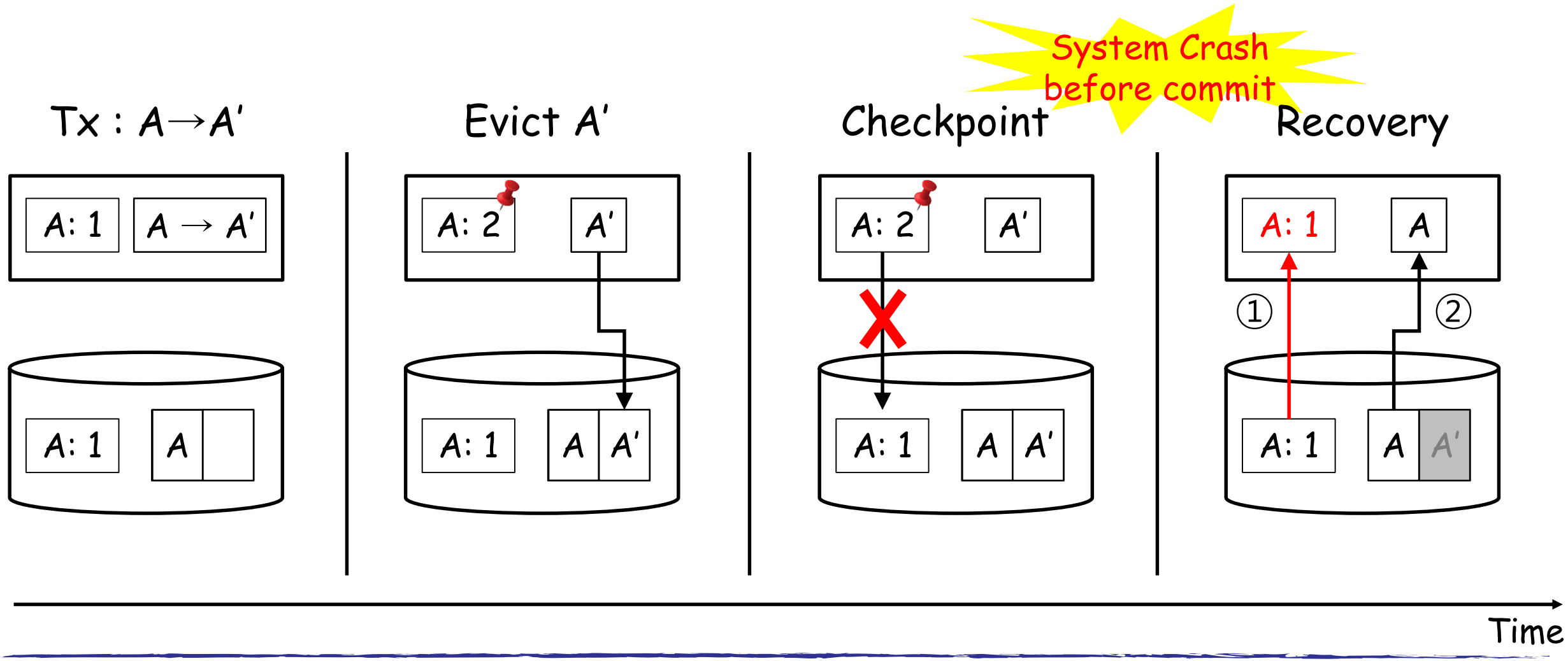


Premature checkpoint of updated mappings



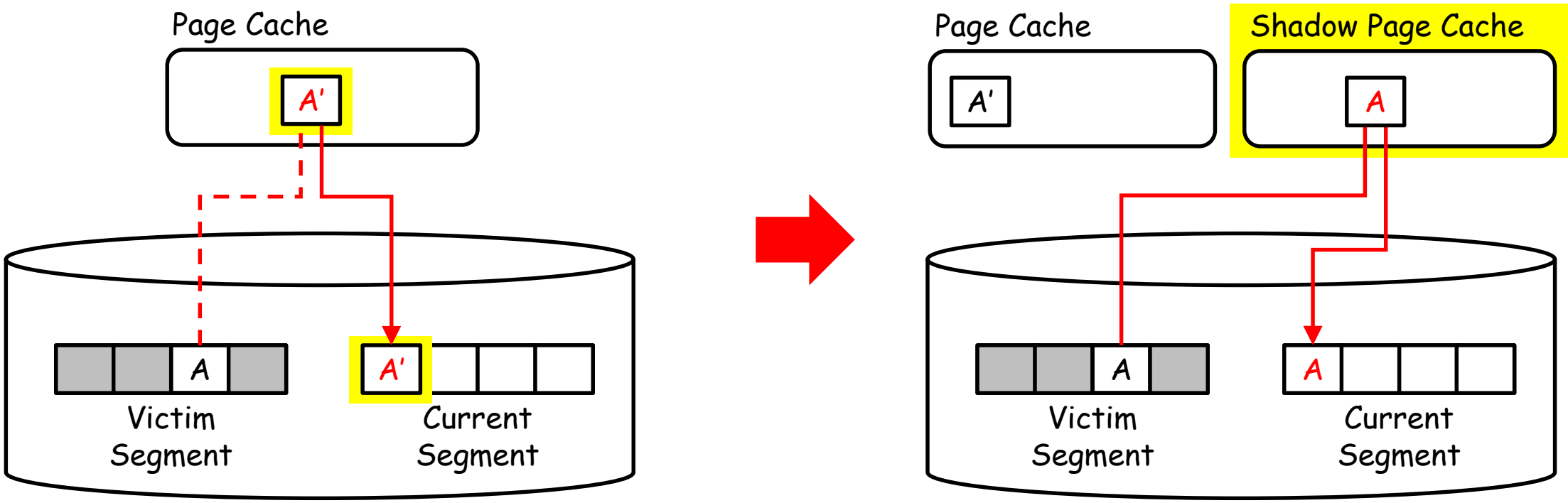
Node Page Pinning

Pin the updated node pages until the transaction commits.



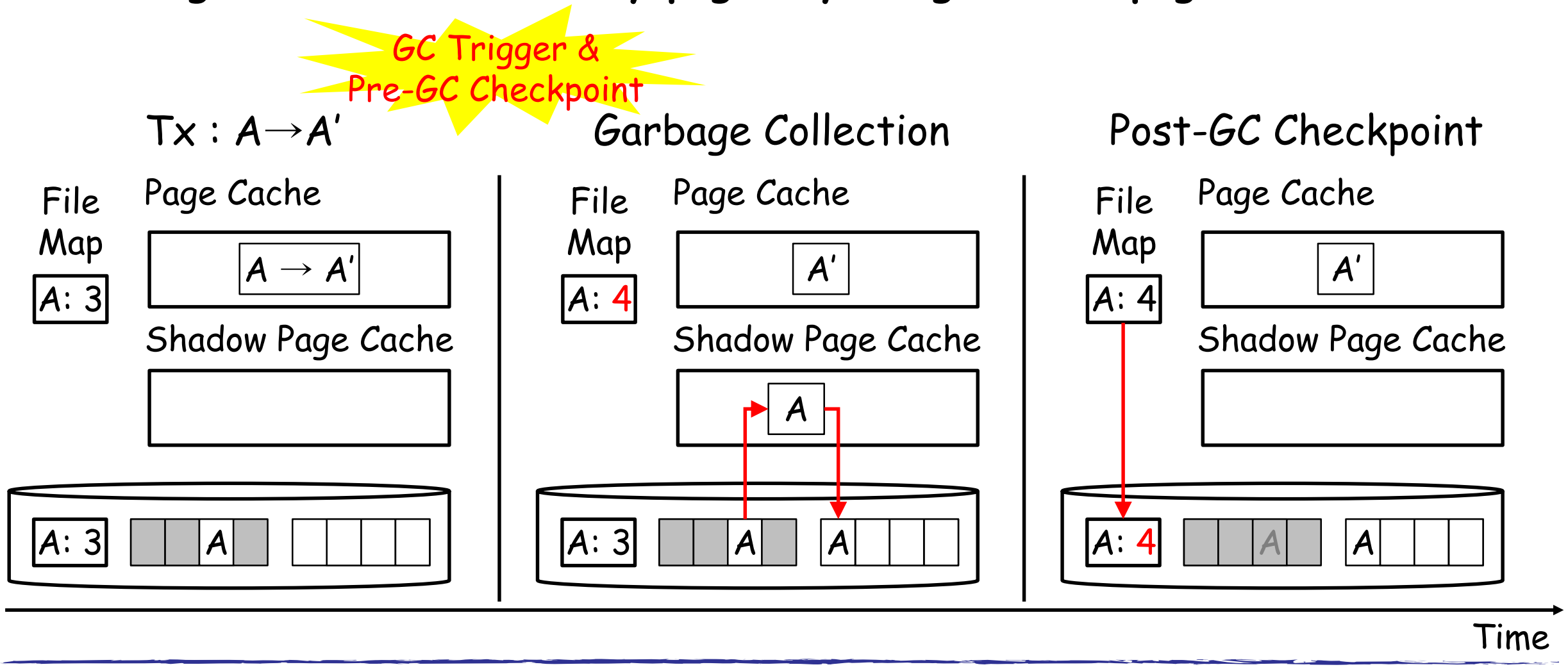
Shadow Garbage Collection

Shadow Page Cache

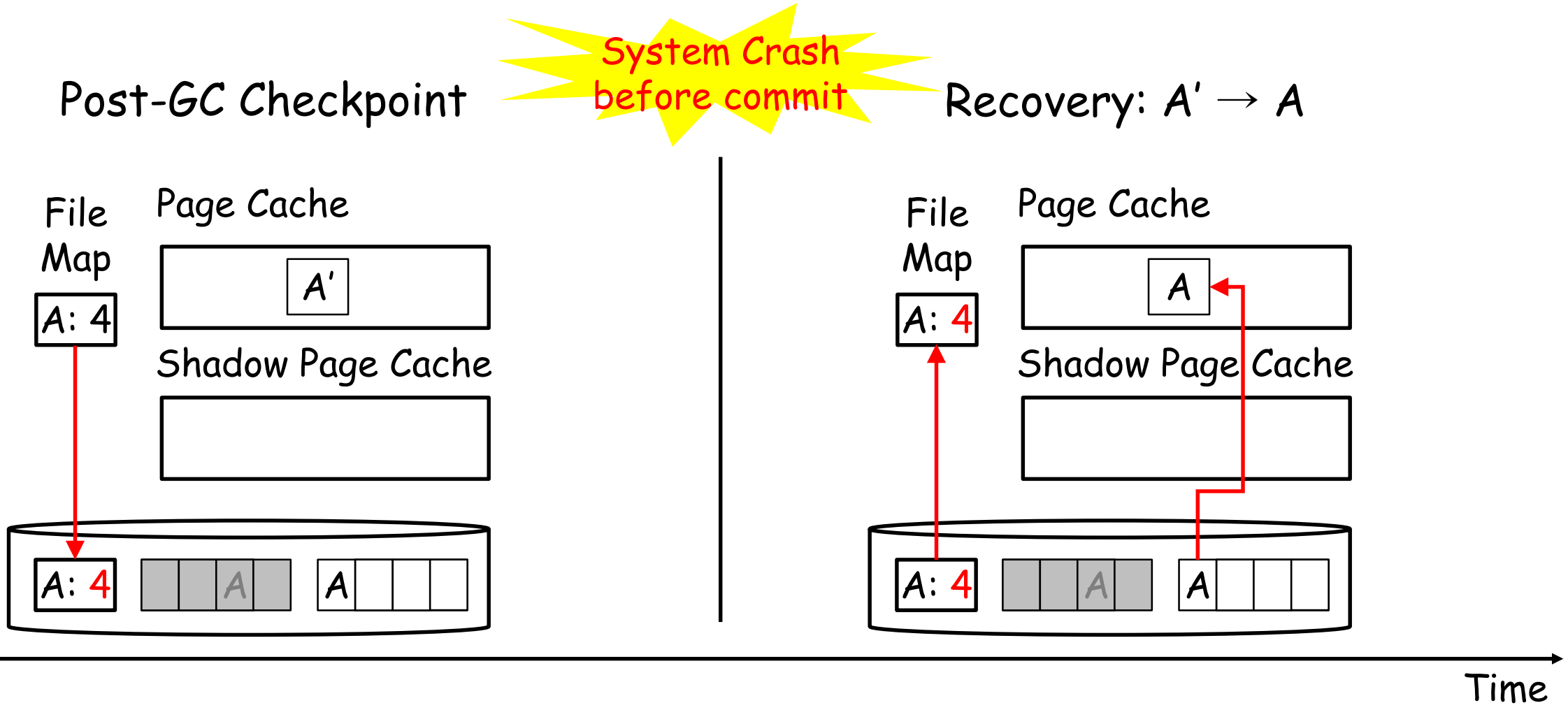


Migration with Shadow Garbage Collection

Migrate transaction dirty pages by using shadow page cache.



Migration with Shadow Garbage Collection (Cont.)



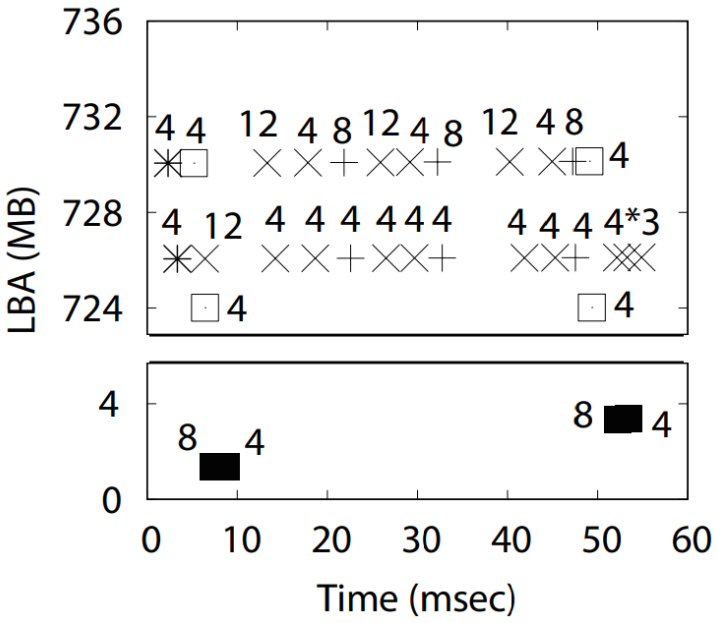
Evaluation

Evaluation

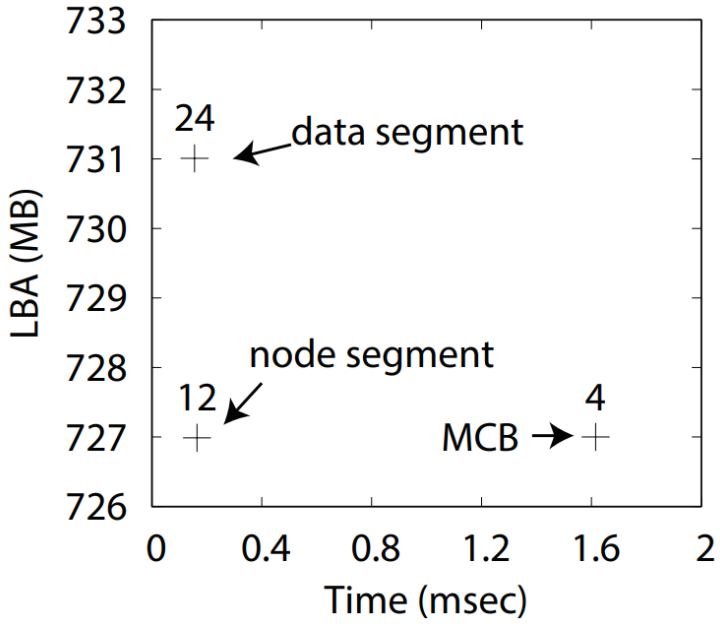
- ✓ HW Setup
  - CPU: Intel CPU i7-9700K (3.60GHz, 4core)
  - Memory: 64GB
  - Storage
    - SATA MLC SSD: Samsung 850 PRO 256GB
    - PCIe 3DXPoint SSD: Intel Optane 900P 980GB
- ✓ Workloads
  - [MoBibench] Multi-database transaction in SQLite
  - [YCSB] Compaction in RocksDB

SQLite with filesystem transaction

- ✓ Tool: MoBibench
- ✓ Operation: Insert, Record Size: 100Byte, # of Tables: 1, # of Databases: 3



F2FS w persist mode



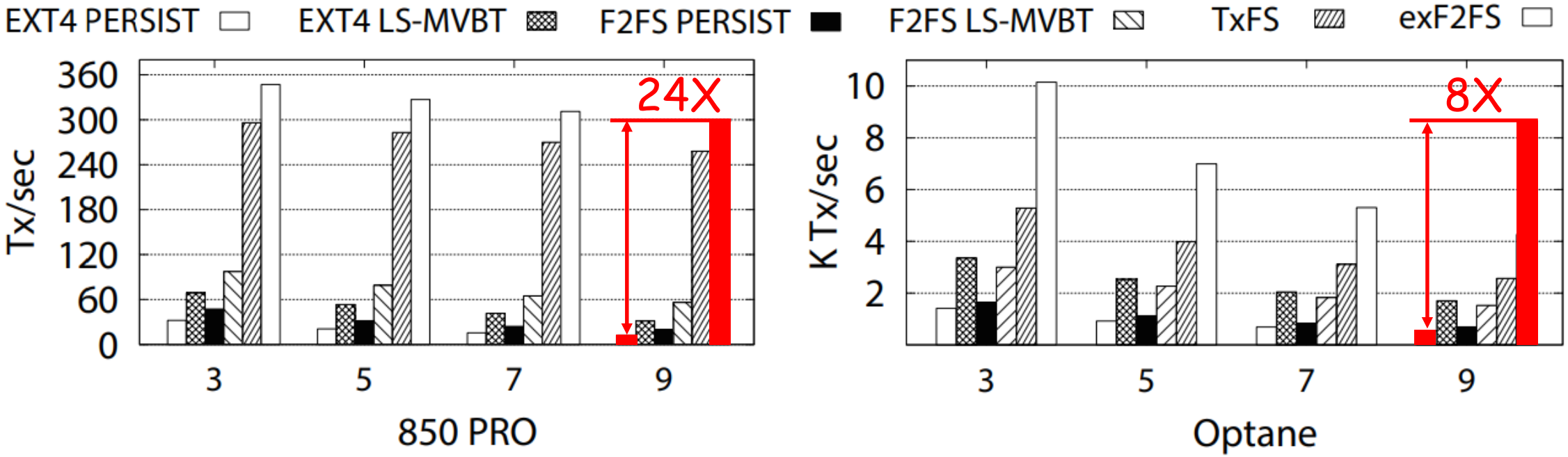
exF2FS

Write Volume: 1/45

Tx Latency: 1/30

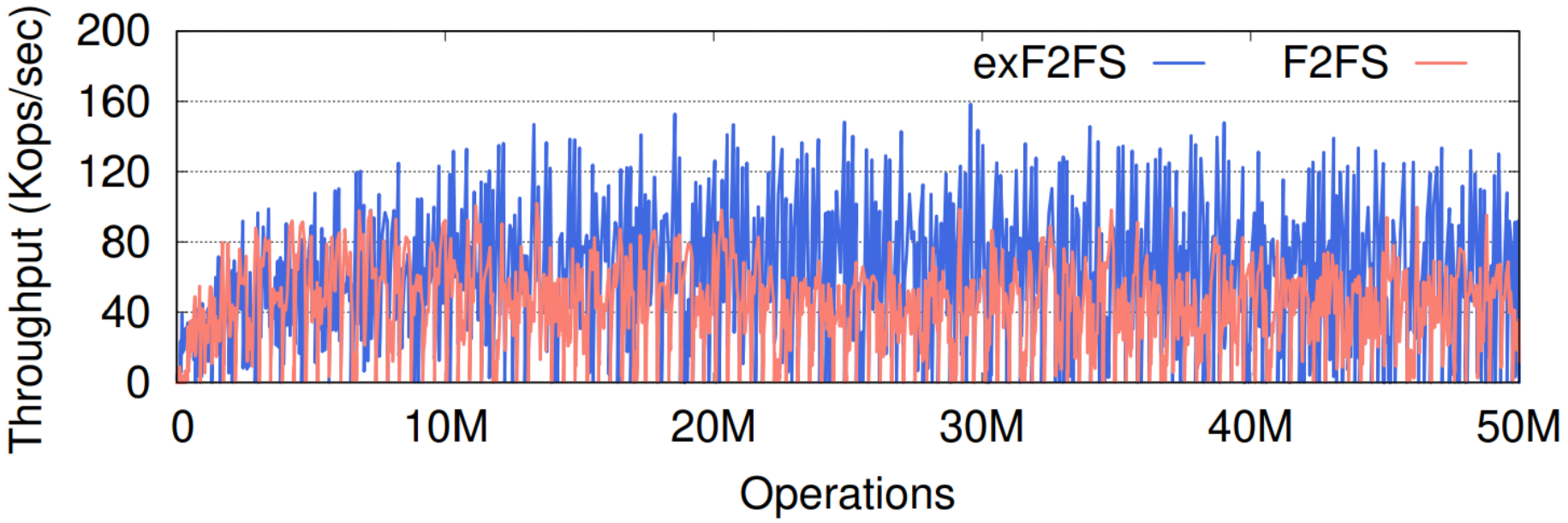
SQLite Throughput

- ✓ Tool: MoBibench
- ✓ Operation: Insert, Record Size: 100Byte, # of Tables: 1, # of Databases: {3, 5, 7, 9}



RocksDB Compaction

- ✓ Tool: YCSB (Workload-A)
- ✓ Memtable size: 64MB, Record size: 1KB, # of operations: 50M



exF2FS decreases the tail latency (99.99%) of the compaction by 87%

Conclusion

- ✓ We developed a new transactional filesystem based on F2FS: exF2FS
- ✓ exF2FS solves three limitations of existing transactional filesystem
  - ➢ Multi-file transaction → Membership-Oriented Transaction
  - ➢ Large size transaction support → Stealing
  - ➢ Compatibility with garbage collection → Shadow Garbage Collection
- ✓ exF2FS improves SQLite performance as 24X and decreased the compaction tail latency of RocksDB by 87%

https://github.com/ESOS-Lab/xF2FS

Question & Answer