Can Applications Recover from **fsync** Failures?

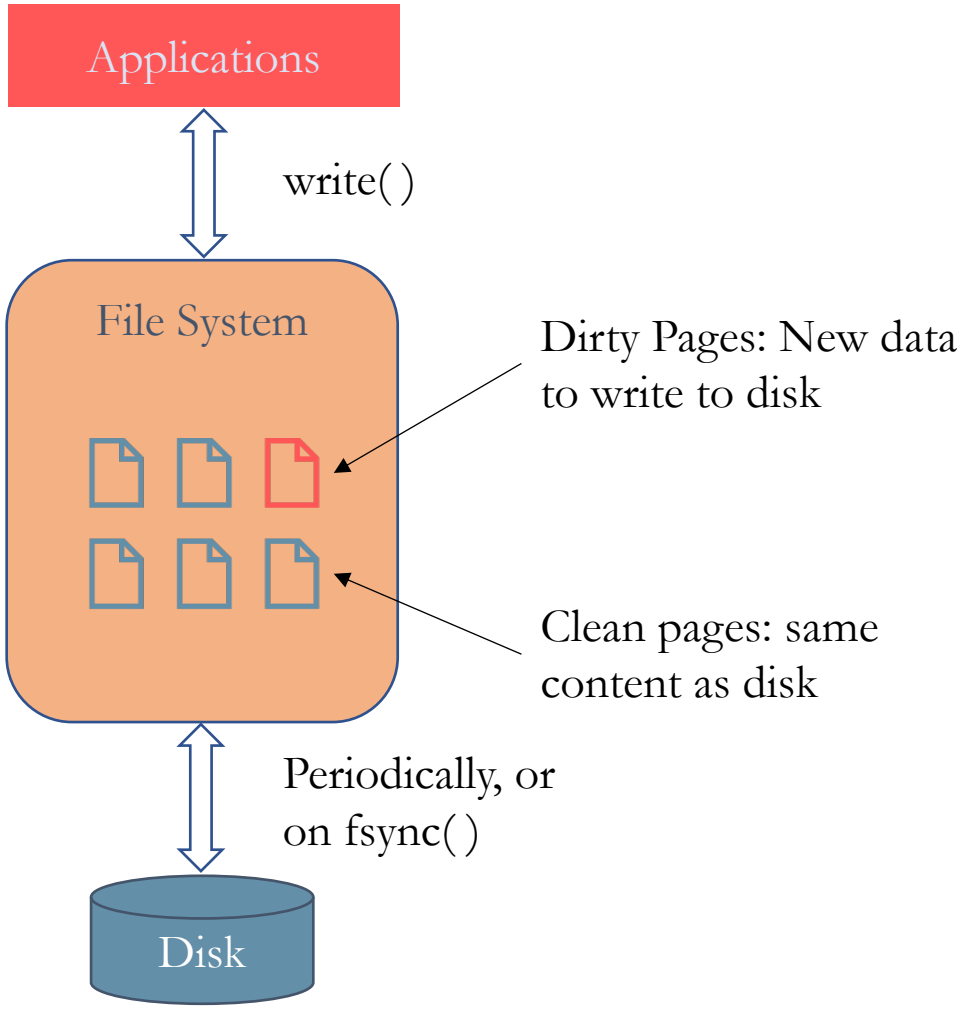
Anthony Rebello, Yuvraj Patel, Ramnatthan Alagappan,
Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau
University of Wisconsin-Madison

Motivation

"How is it possible that PostgreSQL used fsync incorrectly for 20 years?"

- Tomas Vondra, FOSDEM 2019

Why do we use fsync?



Applications use the file system to access the disk

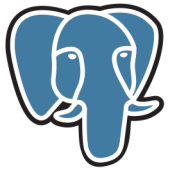
Writes are usually buffered in the page cache

- Modified pages are marked dirty

Dirty pages can be written immediately using fsync( )

Who uses fsync?

Applications that care about durability



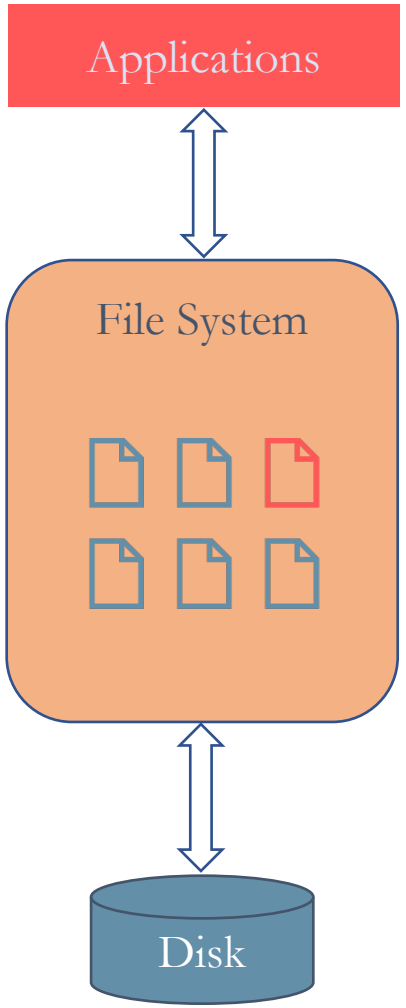
PostgreSQL

LMDB

Support many file systems

- Ext4, XFS, Btrfs
  - Different design and implementation
  - Standard interface

Fsync can fail



File system interacts with disk during fsync

Disks can fail

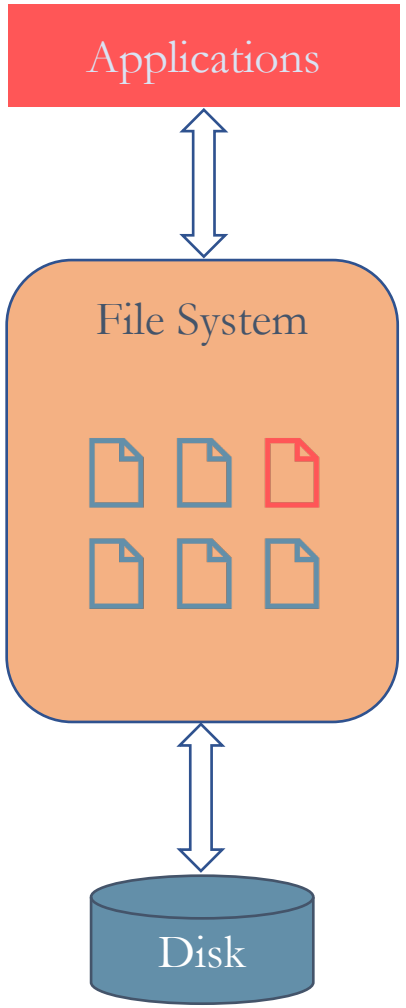
- Latent sector errors, transient failures over network

File system returns -1 and errno set to EIO

Difficult for applications to store data correctly

Our work

Systematic study of the storage stack

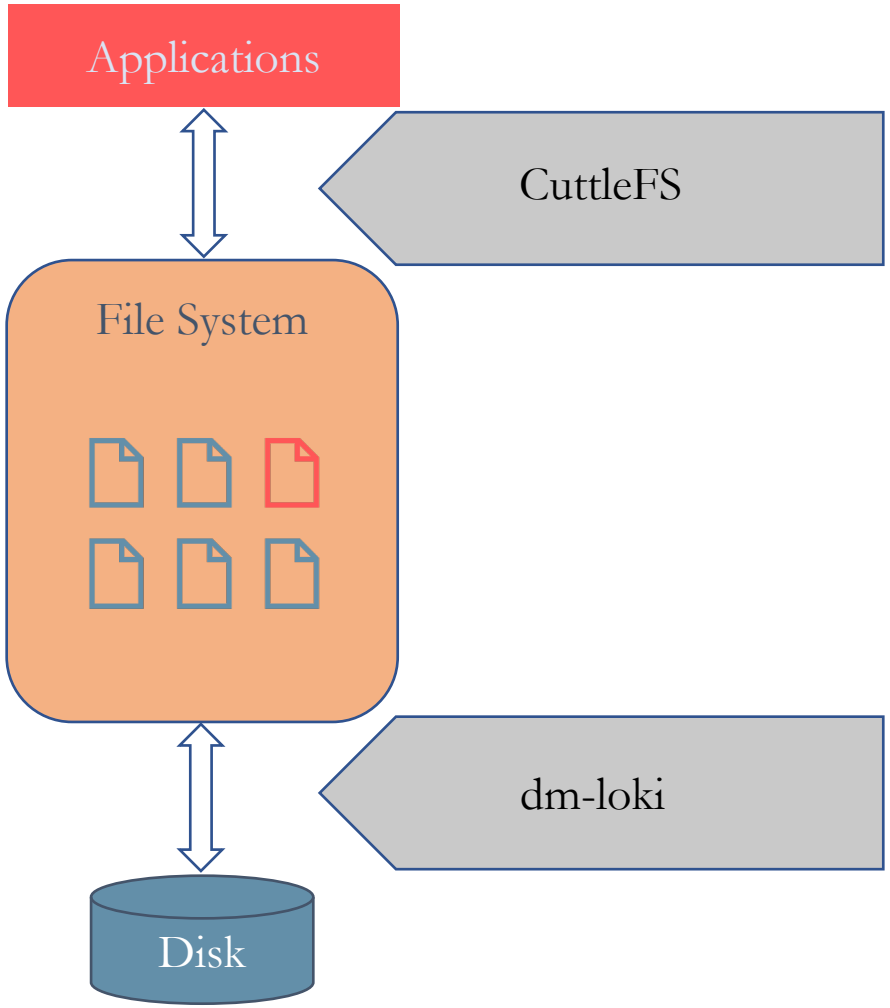


2 **Application reactions** to fsync failures
- Redis, LMDB, LevelDB, SQLite, PostgreSQL

1 **File system reactions** to fsync failures
- Ext4, XFS, Btrfs

Methodology

Inject faults and study reactions



2 Intercept all write/read/fsync calls

- FUSE file system – CuttleFS
  - Fail certain requests
  - Behave like ext4/XFS/Btrfs after failing

1 Intercept all block requests that go to disk

- Custom device mapper target – dm-loki
  - Fail certain requests

Results

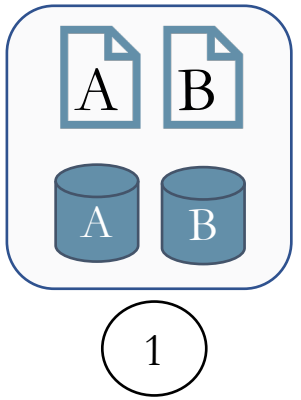
File Systems (2 results)

Applications (2 results)

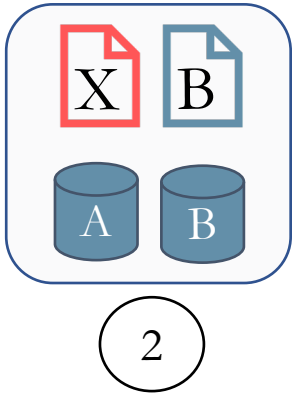
File System Result #1: Page Contents Differ

File systems do not handle fsync failures uniformly

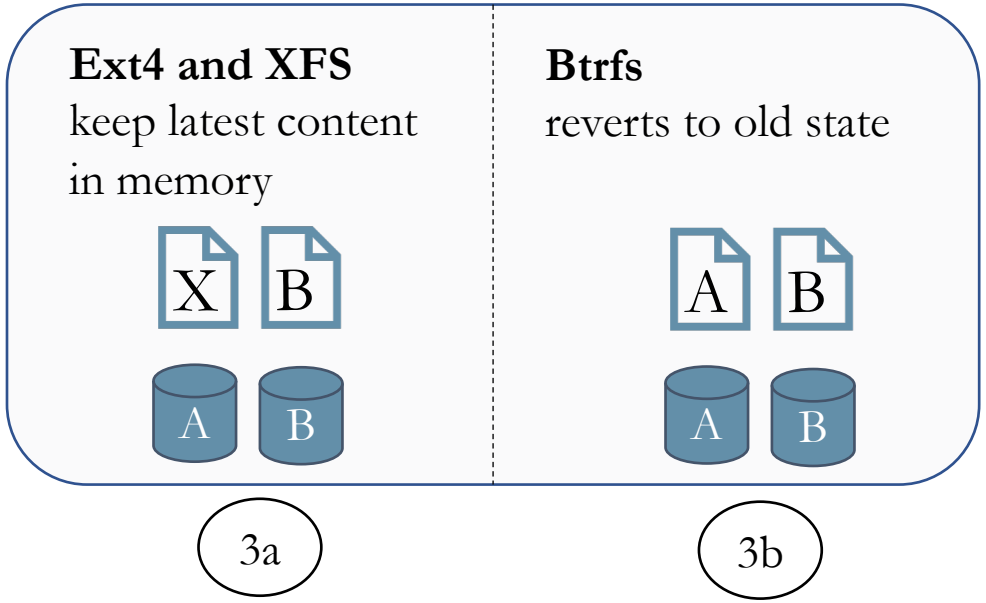
- Example: Page content after fsync failure



Changing A to X dirties the page in memory



fsync( ) fails to write X to disk
Contents in memory depend on the file system.

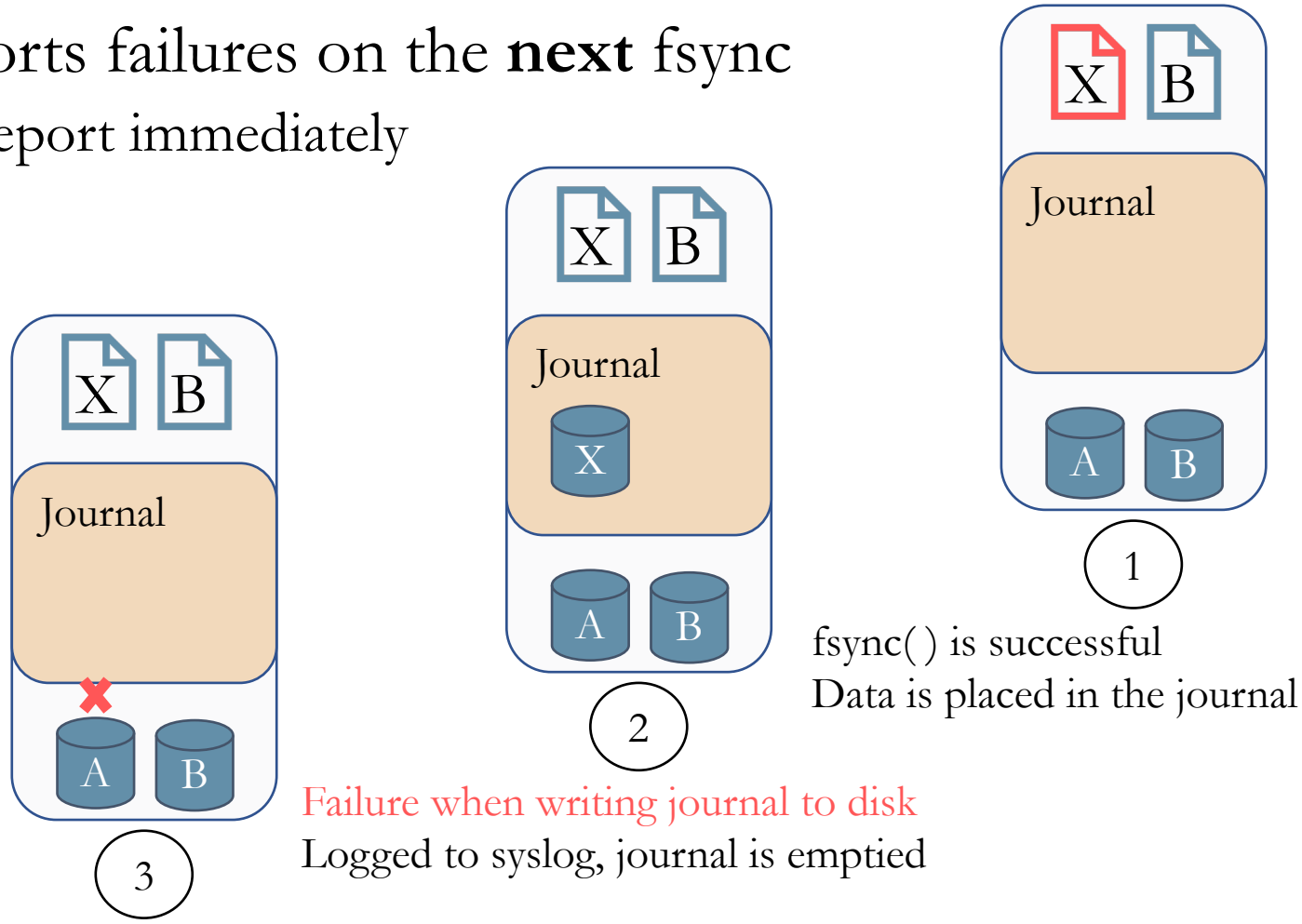


Applications cannot trust page cache contents

File System Result #2: Delayed Reporting

Ext4 data mode reports failures on the **next** fsync

- Other file systems report immediately

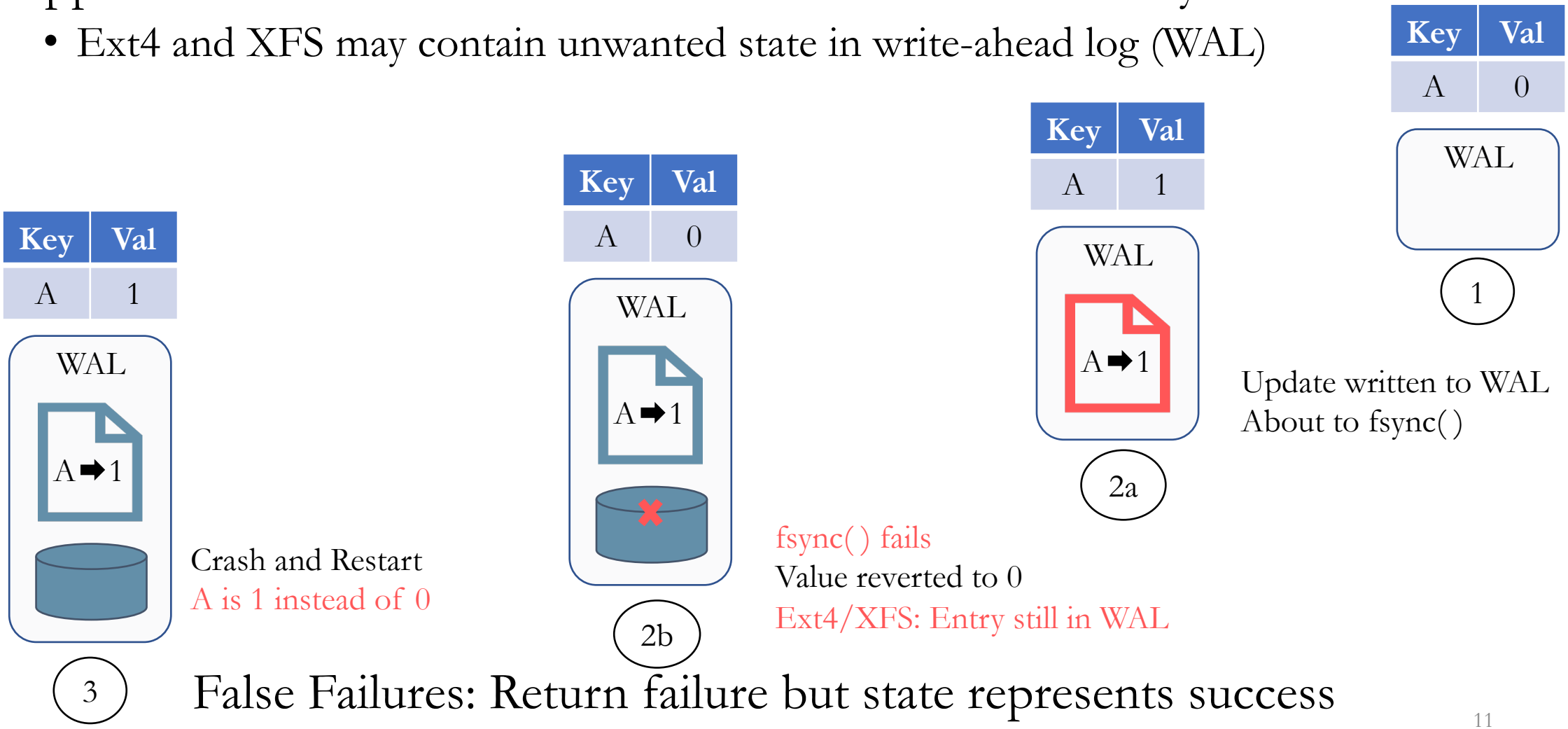


Confusing for applications

Application Result #1: Incorrect Recovery

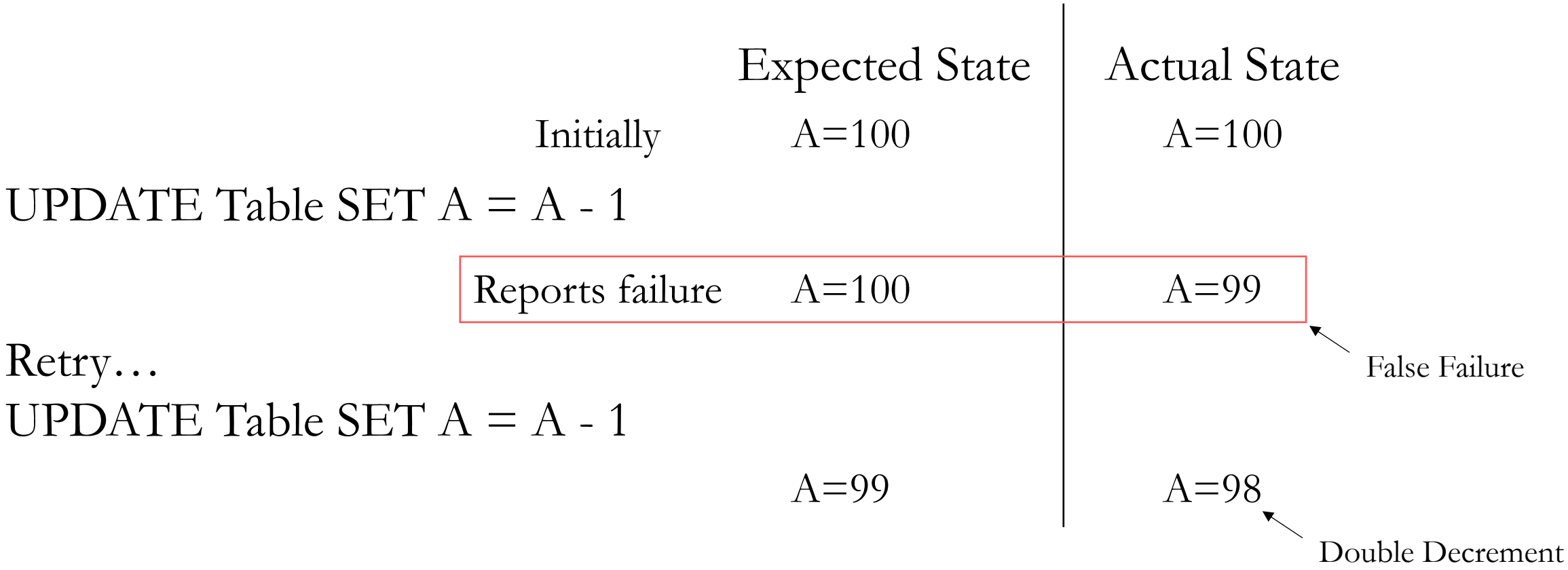
Applications that crash and restart recover state incorrectly

- Ext4 and XFS may contain unwanted state in write-ahead log (WAL)



False Failures: Return failure but state represents success

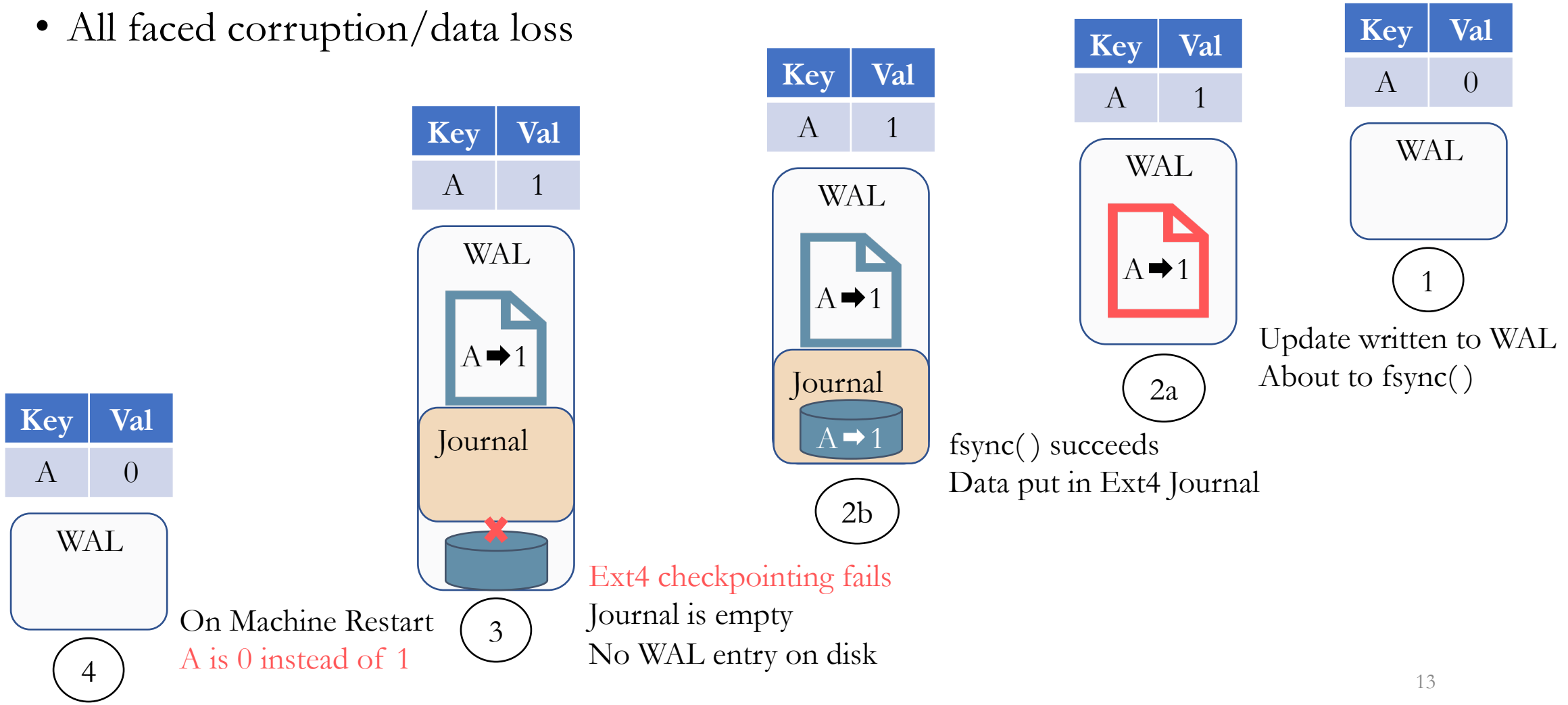
Application Result #1: False Failures



Application Result #2: Corruptions & Data Loss

Applications defenseless against ext4 data mode delayed error reporting

- All faced corruption/data loss



Conclusion

File systems do not handle fsync failures uniformly

- Page content depends on file system
- Error reporting is not always immediate

Applications do not handle fsync failures correctly

- Incorrect recovery from page cache
- Defenseless against late error reporting

Can applications recover from fsync failures?

- Maybe, if …
  - Developers write file-system specific code

Need to standardize file system behavior for fsync failures

Questions?

Anthony Rebello
arebello@wisc.edu

Thank You