Optimistic Crash Consistency

Vijay Chidambaram, Thanumalayan Sankaranarayana Pillai, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

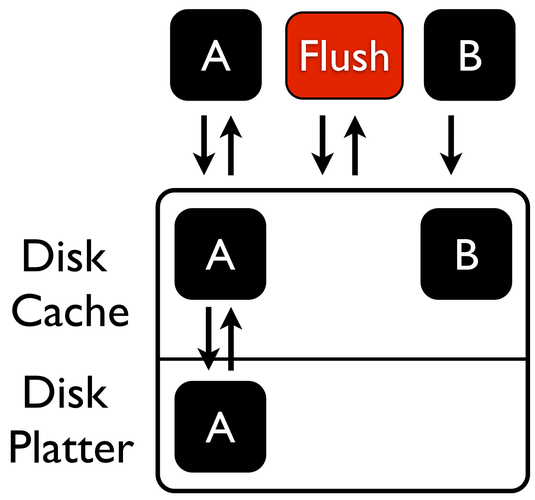
WISDOM

Crash Consistency

- Modern file systems maintain crash consistency by carefully ordering writes to disk
- File system conflate ordering writes to disk with durability, thus making ordering very expensive
- Maintaining consistency degrades performance by 10x for some workloads
- Users forced to choose between performance and consistency

Ordering Disk Writes

- Problem: disk writes are ordered with expensive cache flushes
- Inefficient when only ordering is required

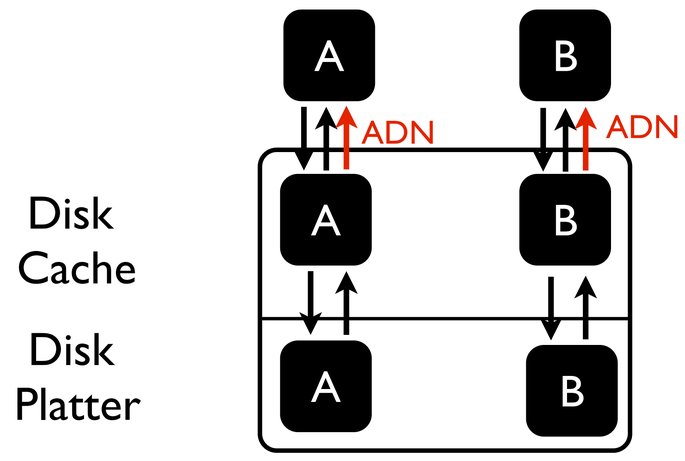


Optimistic Crash Consistency

- Provides both high performance and strong crash consistency
- Decouples ordering from durability
- Eliminates flushes in the common case
- Employs checksums, delayed writes, and other techniques
- osync() provides ordering among writes at high performance and eventual durability

Asynchronous Durability Notifications

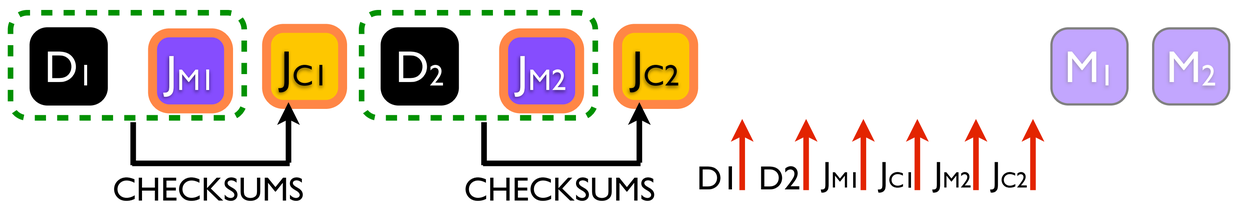
- Extra signal to upper layer when block is destaged from cache to platter
- Frees disk to optimize writes for maximum efficiency



Optimistic Journaling

- Ext4 journaling uses disk cache flushes between different phases of journaling to ensure ordering among disk writes *

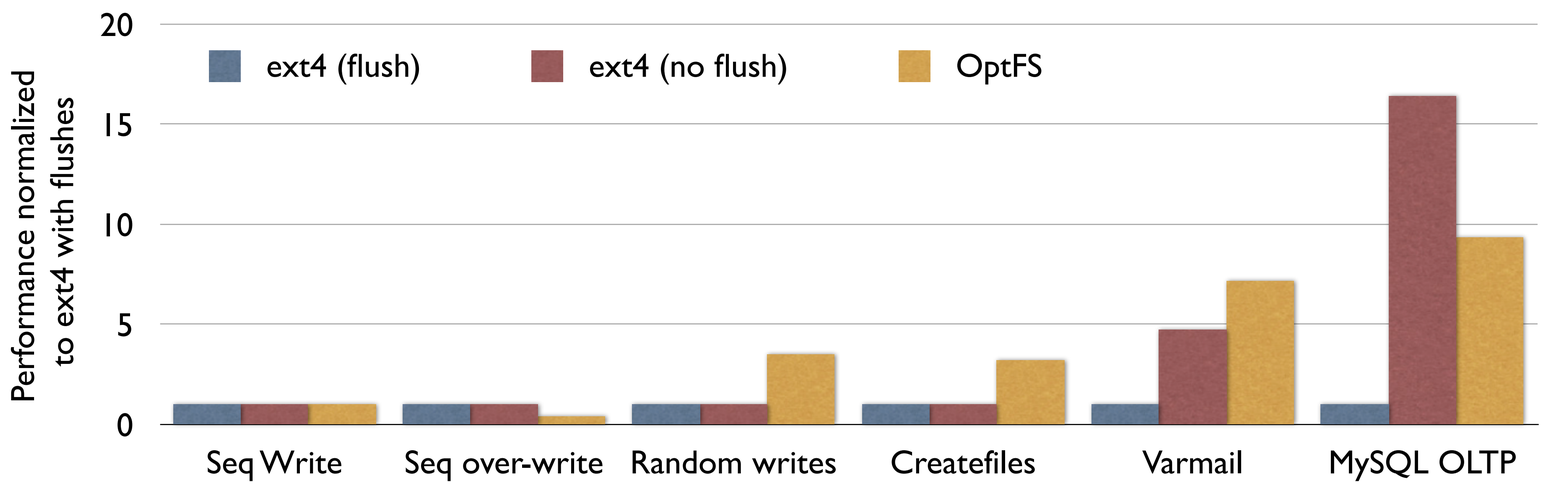
- Optimistic journaling removes those flushes, and handles the resultant re-ordering of blocks using different techniques



- Checksums are used to detect if the journal commit (J_C) is reordered before D and J_M
- The metadata write (M) is delayed until durability notifications (↑ADN) are received for the previously issued D, J_M, and J_C (metadata writes happen in the background)

* D Data | J_M Journal metadata | J_C Journal commit | M In-place metadata

Performance Evaluation



- We compare the performance of OptFS against ext4 with and without flushes
- OptFS performs 3-10x better than ext4 with flushes on many workloads
- OptFS performs almost as well (and sometimes better) than ext4 without flushes, despite providing strong consistency

Case study: SQLite using osync() on OptFS

	Ext4 w/o flush	Ext4 w/ flush	OptFS
Inconsistent	73	0	0
Old state	8	50	76
New state	19	50	24
Time/op (ms)	23.28	152	15.3

- Crashed SQLite in middle of transactions
- Studied behavior after recovery
- Using osync(), SQLite provides ACI (with eventual durability) semantics at 10x the performance of ext4 with flushes