Reclaiming Good Transactions from a Corrupt Journal

Computer Science
UNIVERSITY OF TORONTO

Shehbaz Jaffer, Bianca Schroeder

Motivation

Persistent storage media is becoming larger and denser.

There is increased media corruption and hardware unreliability.

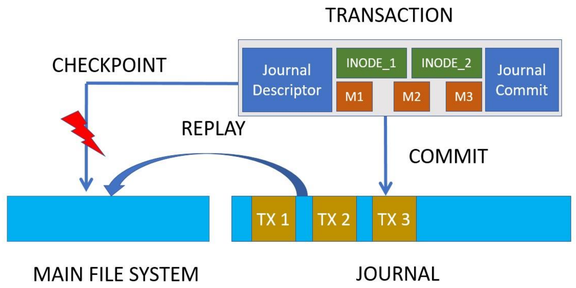
Storage systems need robust recovery techniques.

Current File System recovery techniques - replication and journaling.

Journaling

Journaling removes time consuming ***fsck*** run on unclean shutdown.

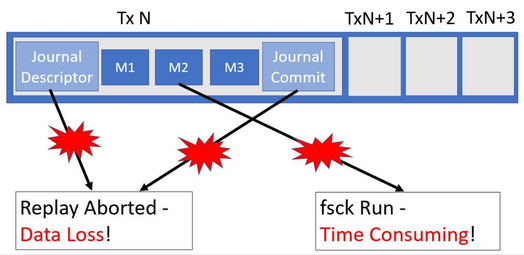
It also writes metadata to a linear log which increases performance.



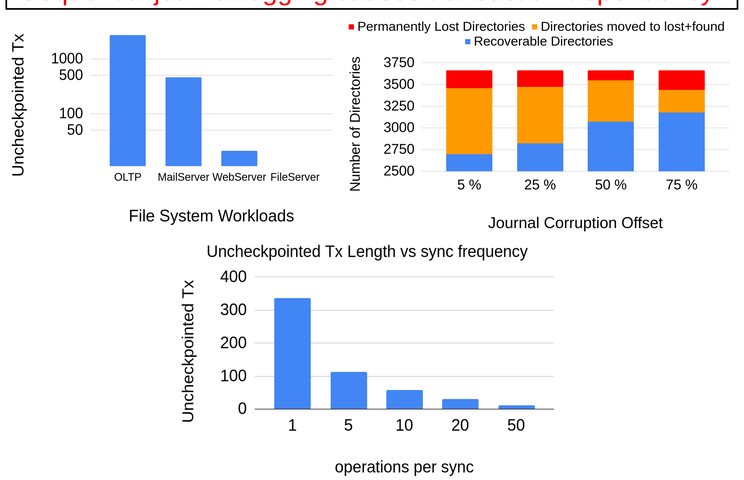
Problem

Journal header or Journal commit corruption leads to data loss.

Journal metadata corruption leads to time consuming ***fsck*** run.



Sequential journal logging causes transaction dependency!



Solution: Sub-Journaling

Log atomic file system updates into different sub-journals.

Each sub-journal can be replayed ***independent*** of the other sub-journal.

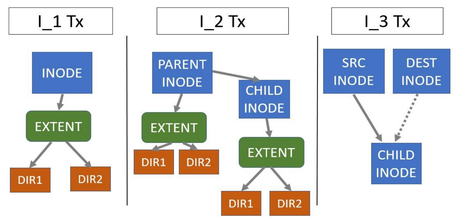
File system remains consistent after any sub-journal replay.

Categorize Transaction Updates

Number of Inodes updated

File System operation	0 inodes (I_0 Tx)	1 inode (I_1 Tx)	2 inodes (I_2 Tx)	3 inodes (I_3 Tx)
	mount	truncate	create	rename
	access	chmod	rmdir	
	stat	chown	mkdir	
	chroot	utimes	link	
	chdir	read	symlink	
	open	write	unlink	

Inodes updated by same transaction have parent-child relationship



Invariants Maintained

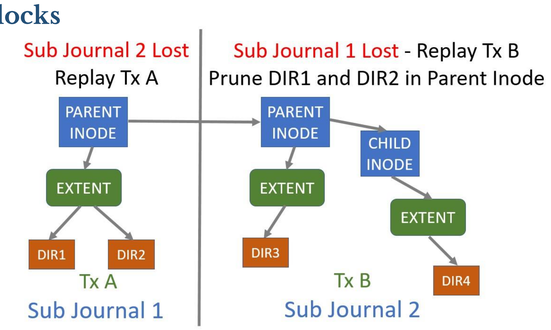
Map all metadata of one transaction to one sub-journal.

Map metadata of one inode to one sub-journal.

Remap parent inode to child inode's sub-journal for I_2 Tx and I_3 Tx.

Recovery after Crash

Inode Blocks



Extent and Directory Block

Prune ***extents*** containing lost directories.

Group Descriptor and Bitmaps

Regenerate after restoring other metadata from available sub-journals.

Deleted Inodes

re-allocation is delayed until checkpoint.

Orphan list is sent to each sub-journal.

Superblock

Record in each sub-journal

Implementation

ext4

1 LOC changed

jbd2_journal_dirty_metadata (inode_no)

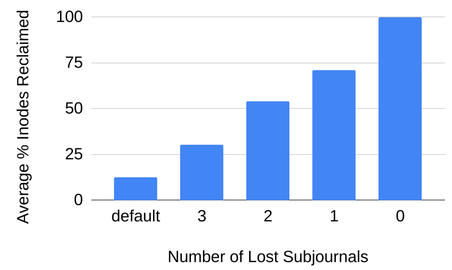
Register Transaction category - I_0 Tx, I_1 Tx, I_2 Tx, I_3 Tx

jbd2

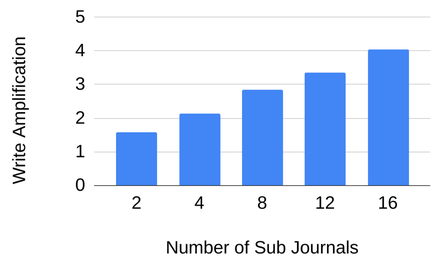
700 LOC added

In Memory Hash Maps	On Disk Structures
<Tx_Handle - Sub-Journal>	Add to Transaction Descriptor Block
<Tx_Handle - Journal Blocks>	<Inode - Sub-Journal> Map
<Tx_handle - Inode>	<Metadata block - Sub-Journal> Map

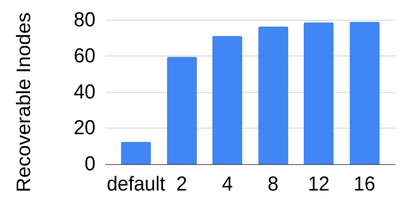
Results



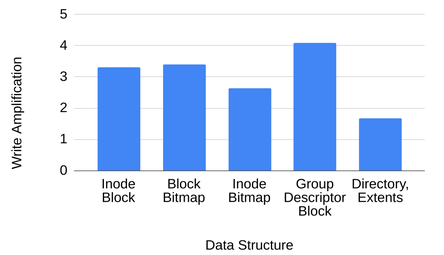
Recoverable inodes for a 4 sub-journal file system with varying lost sub-journals.



Sub-journaling technique incurs Write Amplification with more sub-journals.



Recoverable Inodes after 1 sub-journal loss with different sub-journal configurations.



Write Amplification Data Structure distribution for 4 sub-journal file system.

Future Work

Reduce Write Amplification by optimizing bitmap and descriptor block logging.

Handle delete operations by tracking orphan inodes across sub-journals.

Application crash consistency - journal one application transactions in one sub-journal.

Verify sub-journaling and journaling equivalence.

Related Work

SPANFS – Separates transactions into static domains based on Block Groups.

iJournaling – Fine Grained Journaling focusses on improving fsync() performance.

High Performance Transaction processing aims to improve multi core scalability.

Application Level Crash Consistency provides isolated streams for each application.