Consistency Without Ordering

Vijay Chidambaram, Tushar Sharma, Andrea Arpaci-Dusseau and Remzi Arpaci-Dusseau

Motivation

Current state of the art in file-system crash consistency:

Lazy, optimistic file systems

- Write blocks to disk in any order
- No overhead at run-time
- Need expensive scan after crash
- Example: FFS, ext2

Eager, pessimistic file systems

- Employ ordering points in update protocols
- Constant performance penalty at run-time
- Quick recovery after crash
- Example: NTFS, XFS, ZFS, brtfs

Is the best of both worlds possible?

Performance benefits of lazy approach with strong consistency and availability of eager file systems

Why are ordering points bad?

- Introduce waiting into file-system code
- Constrain the scheduling of disk writes
- Increase complexity, leading to bugs
- Require lower-level primitives like disk cache flush
- SATA/IDE drives known to implement CACHE FLUSH command incorrectly [1,3,4]
- Operating system runs on stack of virtual devices
- If a single layer ignores flush commands, file-system consistency is compromised
- Virtual machines ignore flush commands to batch writes and improve performance
- From VirtualBox [2]:

  "If desired, the virtual disk images can be flushed when the guest issues the IDE FLUSH CACHE command. Normally these requests are ignored for improved performance"

No-Order File System (NoFS)

- Employs backpointer-based consistency
- Uses non-persistent allocation structures

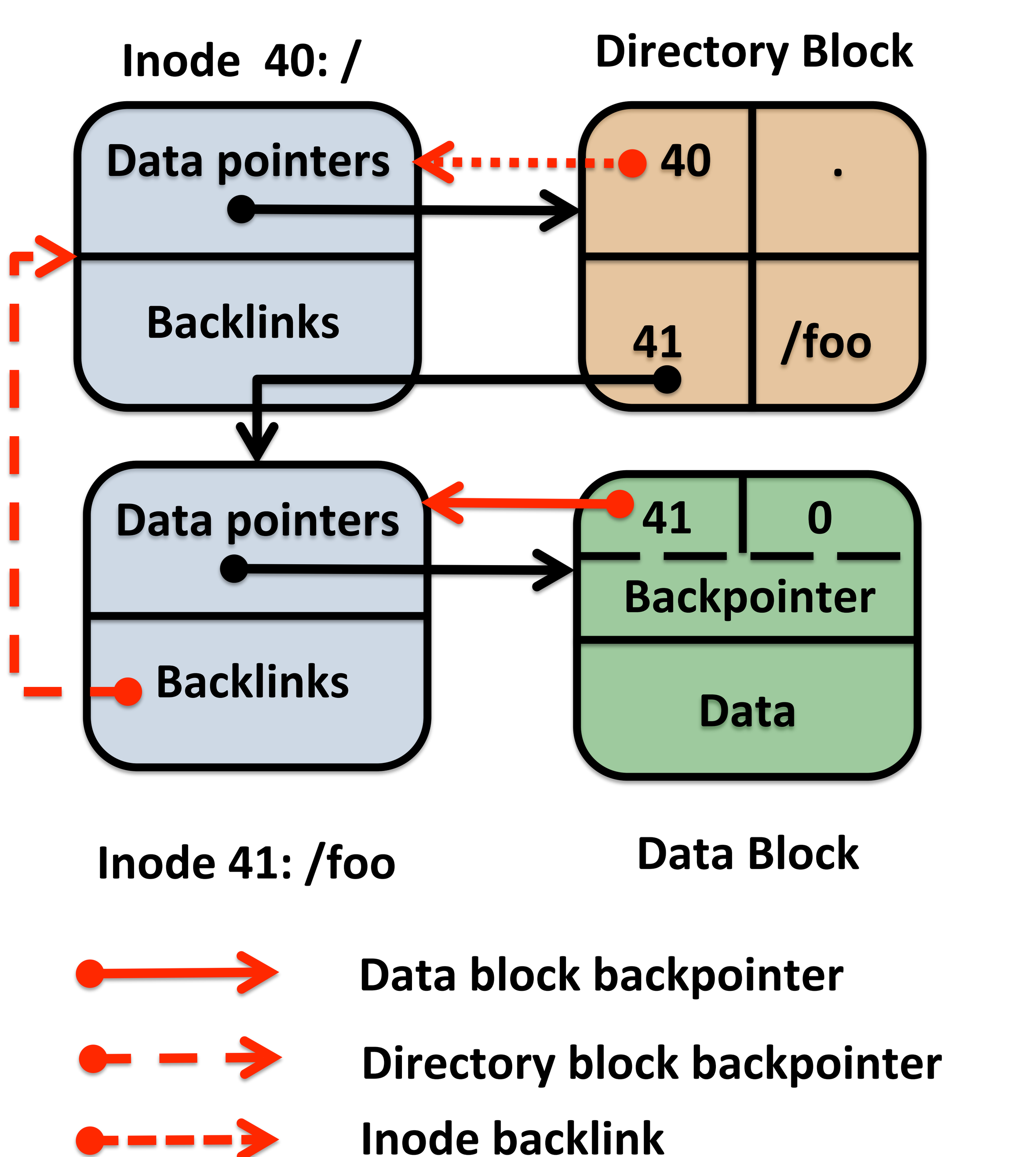
Backpointer-based consistency:

- Associate each object with its logical identity
- Embed a backpointer in each object
- Backpointers identify owner object
- Data block backpointer points to owner file
- File backpointer points to parent directories
- Write blocks to disk in any order
- Use backpointers to resolve inconsistencies

Key Assumption:

- Backpointer and block data written atomically
- Current SCSI drives 520 byte atomic write
- Future disk drives will potentially provide 4K + 8 bytes atomic write

Backpointers in NoFS



Non-persistent allocation structures

- Allocation structures such as bitmaps cannot be trusted after a crash
- Hence, they must be verified before being used
- NoFS does not maintain on-disk allocation structures. Only in-memory versions are used.
- Backpointers allow allocation structures to be recomputed incrementally in the background after file-system mount
- Background threads use backpointers to determine allocation status of each object and accordingly update in-memory structures

References

[1] Abhishek Rajimwale, Vijay Chidambaram, Deepak Ramamurthi, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Coerced Cache Eviction and Discreet-Mode Journaling: Dealing with Misbehaving Disks. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'11)*, Hong Kong, China, June 2011.

[2] VirtualBox Manual. Responding to guest IDE/SATA flush requests. http://www.virtualbox.org/manual/ch12.html.

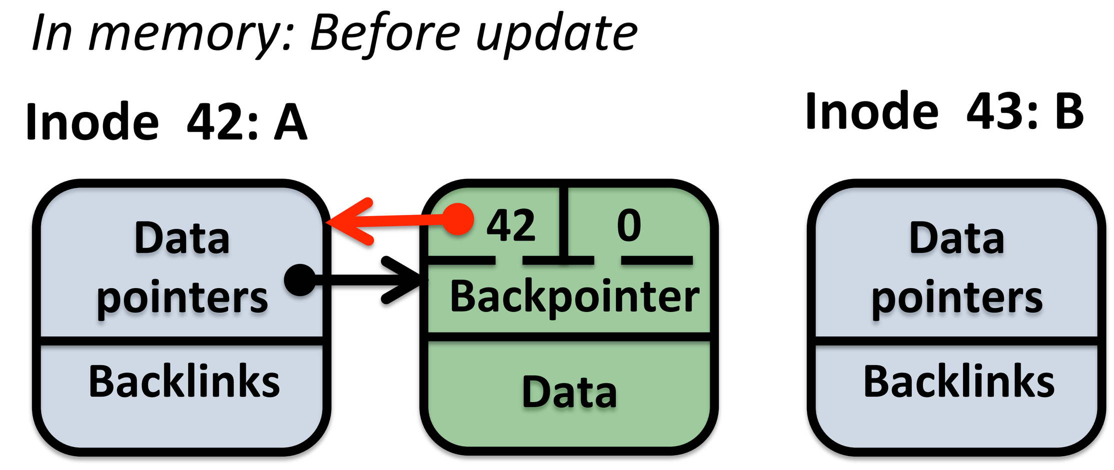
[3] Seagate Forums. ST3250823AS (7200.8) ignores FLUSH CACHE in AHCI mode. http://bit.ly/xcSAUV, September 2011.

[4] R1Soft. Disk Safe Best Practices. http: //wiki.r1soft.com/display/CDP3/Disk +Safe+Best+Practices, December 2011.

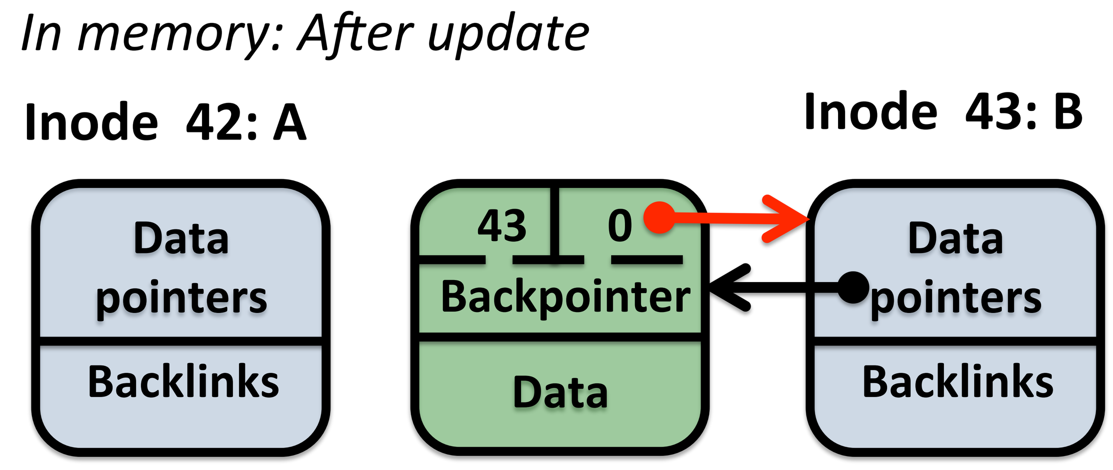
Using backpointers to resolve inconsistencies

Truncate of file A followed by a write to empty file B

In memory: Before update

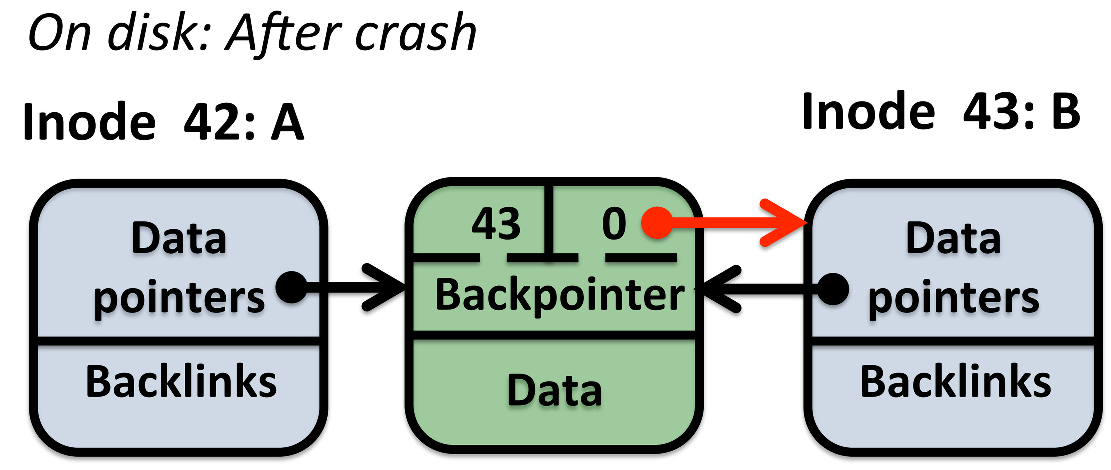


In memory: After update



Due to a crash, only inode B and the data block are updated on disk

On disk: After crash

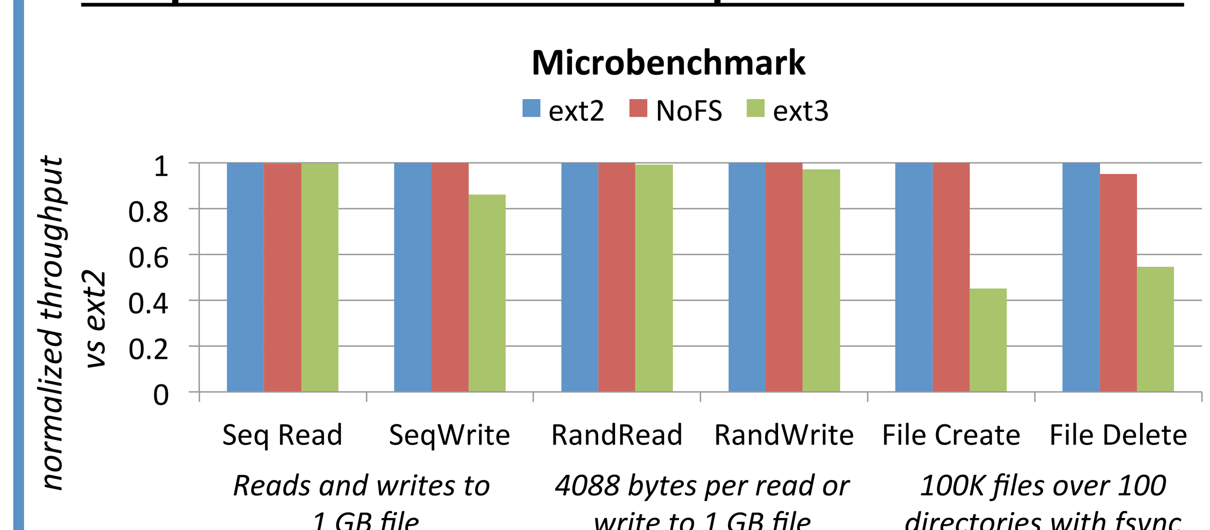


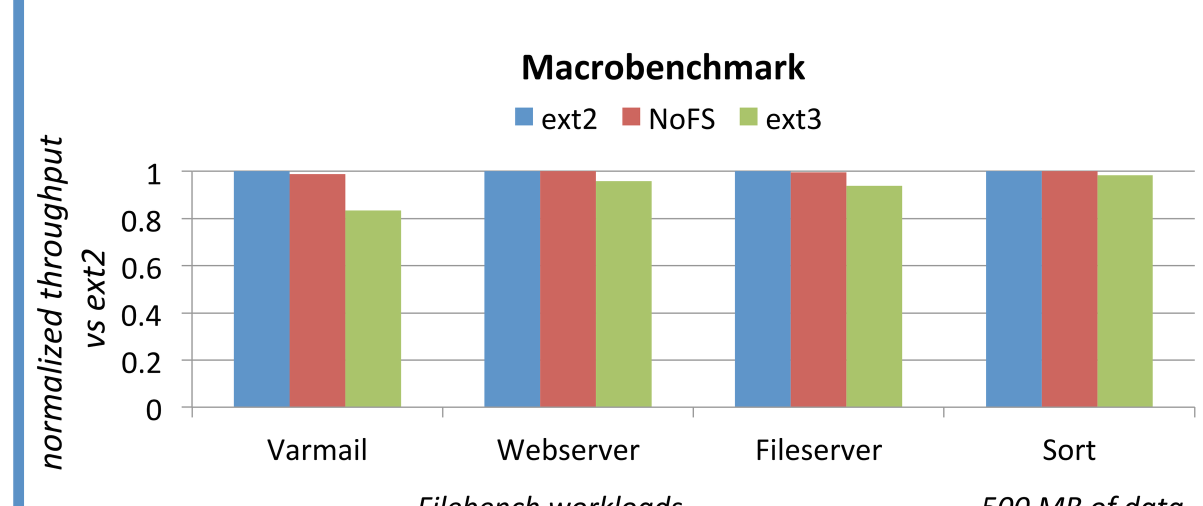
In ext2: This could lead to **data corruption** when A and B both access the data block

In NoFS: When A tries to access the data block, the absence of backpointer is detected and an **error** is returned

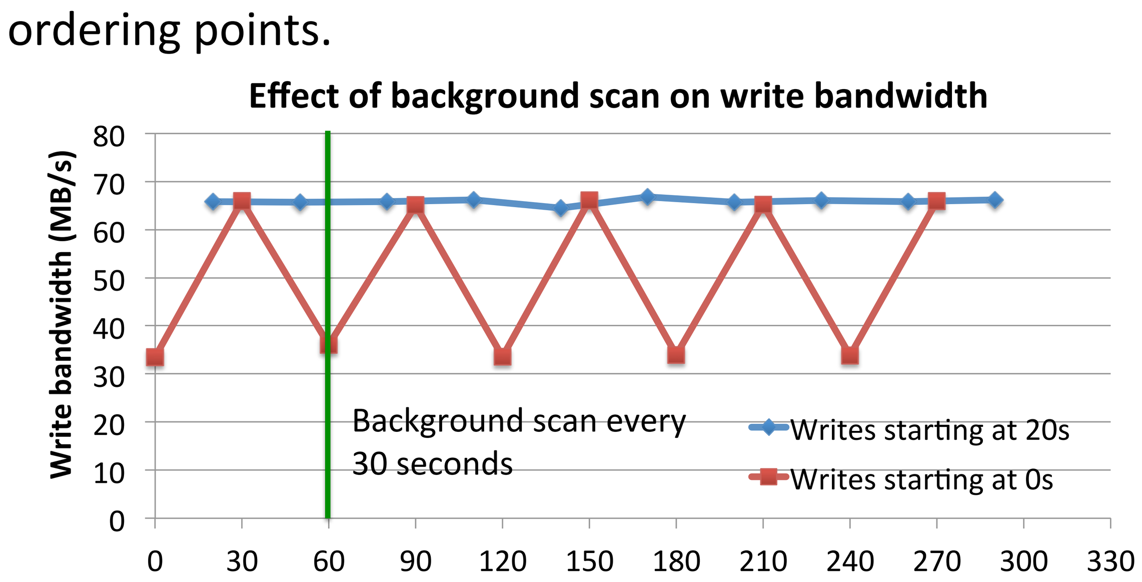
Evaluation

The performance of NoFS compared to ext2 and ext3

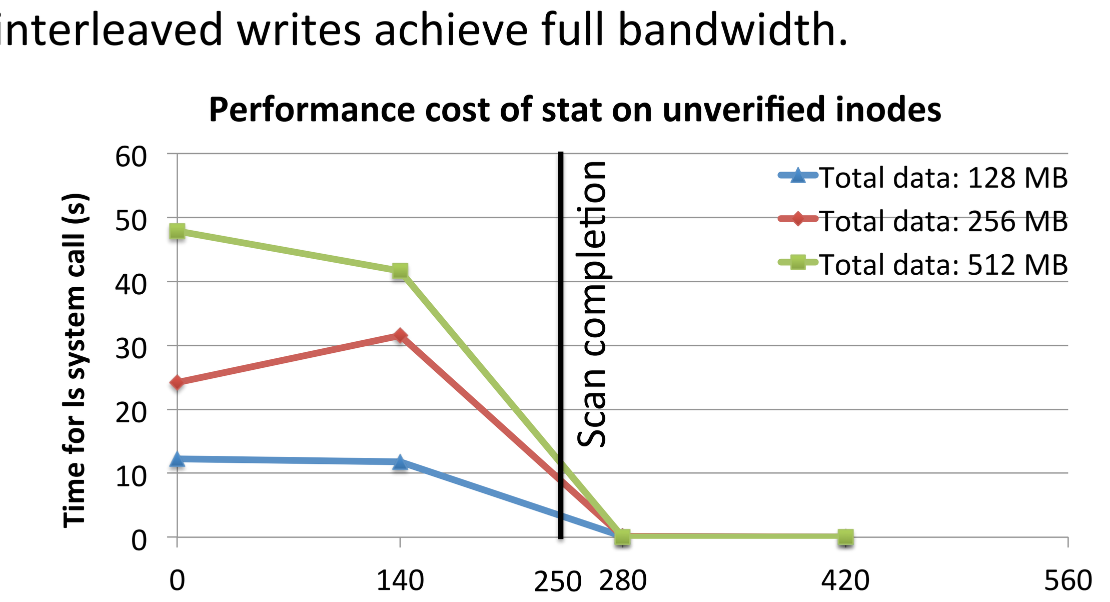




The performance of NoFS is similar to that of ext2 and better than or equal to ext3. File create/delete and varmail workloads (which include a lot of fsync calls) show the performance degradation in ext3 due to ordering points.



When sequential writes are interleaved with the background scan, write bandwidth drops to half. Non interleaved writes achieve full bandwidth.



Running stat on an unverified inode requires checking it by reading all its data blocks; Once the scan is done, all inodes are verified, and this cost is not incurred.