



11/9 (Quiz 3, 10/28-11/9, will include the following two papers)

- Gregory R. Ganger, Marshall Kirk McKusick, Craig A.N. Soules, and Yale N. Patt. "Soft Updates: A Solution to the Metadata Update Problem in File Systems," ACM Transactions on Computer Systems, Vol. 18, No. 2, May 2000, Pages 127-153.
- P. M. Chen, W. T. Ng, S. Chandra, C. Aycock, G. Rajamni, and D. Lowell, "The Rio File Cache: Surviving Operating System Crashes," In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, SIGPLAN Notices 31(9):74-83, September 1996.

Optional further readings on Rio:

David E. Lowell and Peter M. Chen, Free Transactions with Rio Vista, In *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles* October 1997, St. Malo, France, pp. 92.101.

Q: Do you believe the underlying argument in Rio? Why or why not?
Would you use a system that recovered data using Rio?

Purpose of these two papers

- Permanent data crash consistency and integrity
 - If the system crashes in the middle of execution, will data in permanent storage still be consistent, with good data integrity?
- Why still important today?
 - Databases
 - No-SQL databases (e.g. Hadoop)
 - Persistent memory

checkpoint : does it go thru?

Note: Software company - Amazon
traditional company - Walmart

what's the difference.

Meta

- Great example of tech transfer.
 - Greg did this when he was a grad student at Michigan. Kirk, from BSD FFS fame, saw it and in BSD.
- Now all flavors of BSD OSes support it.
 - If you have access to a BSD OS, try deleting a large directory tree and you'll get an immediate sense of the difference.

inode : only data write back
inode are sync

Write Example

Deleting a file

```

graph LR
    abc[abc] --> i1[i-node-1]
    def[def] --> i2[i-node-2]
    ghi[ghi] --> i3[i-node-3]
  
```

Assume we want to delete file "def"

Deleting a file

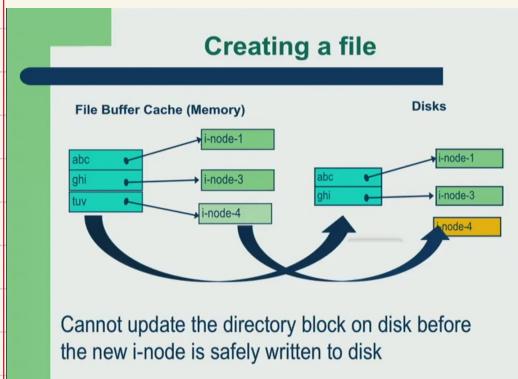
Seq : 1->2

cannot remove
i-node on disk (or mark as free)
before update the directory block to remove def entry

Deleting a file

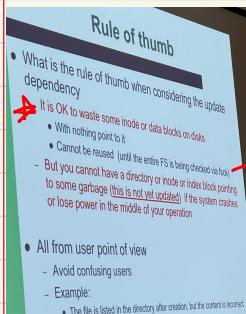
- Correct sequence is
 1. Write to disk a data block with "def" deleted
 2. Write to disk a node block containing deleted i-node (mark as invalid or free)
- Leaves the file system in a consistent state

Creating a File



never have invalid pointer.

Rule of thumb



→ these block will be invalid until recheck.

Problem Recap

- What is the problem?
 - FSEs typically use sync writes to ensure consistency of updates to metadata
 - Solve "update dependency" problem
- What is an update dependency?
 - e.g., create new file: create inode + update directory
 - inode needs to be on disk before entry created in dir
 - If write dir first, then a create immediately after would leave a dangling pointer (invalidated inode)
 - If this sounds like it would happen rarely, think again
- What are the ordering constraints?
 - 1) prevent dangling pointers (e.g., dir before inode)
 - 2) never reuse before removing all pts (e.g., reallocating disk blocks)
 - 3) never reset last pte before creating another (e.g., renaming)

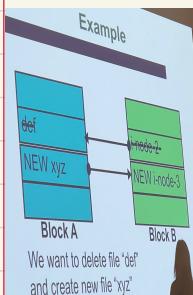
10

11/9/21

CSE221 - Operating Systems,
Yunyuan Zhou

Ex.

circular deps

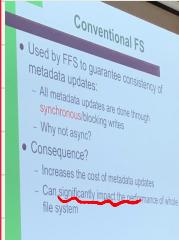


we will never in such situation.

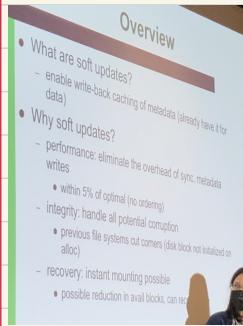
we do this sync, if delete

do it first, do one at a time

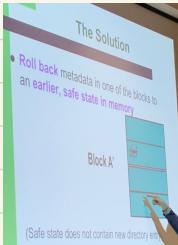
main point



performance penalty.



Algs to calculate state.



- Write first block with metadata that were rolled back (block A' of example)
 - Write blocks that can be written after first block has been written (block B of example)
 - Roll forward block that was rolled back
 - Write that block
 - Breaks the cyclical dependency but must now write twice block A



- Step 3:
• Roll forward Block A and write it to disks

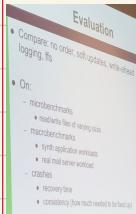


def

NEW xyz

Block A





try do → fault injection → use bad harddrive?



→ added later.

Pure Academic Paper no application

RIO

Overview

- What is the goal of the Rio file cache?
 - the performance of main memory with the reliability of disk
- Why can Rio perform better?
 - eliminates reliability-induced writes to the disk (synchronous writes)
 - recall Ousterhout, LFS papers
- Why is it still reliable?
 - Rio protects file cache memory during normal operation
 - restores file cache contents on "warm" reboot

Warm Reboot

- What is a warm reboot?
 - power not lost, system just reset
- What is a key requirement of a warm reboot?
 - RAM not reset
 - note that not all architectures support this

- persistent memory.

Design & Implementation

- How does Rio protect the file cache in the name?
 - VM support
 - makes file cache pages read-only
 - need to ensure that all accesses to those pages use VM protection
 - DEC Unix by default does not do this (uses phys addrs)
 - when pages need to be written, they are temporarily unprotected
 - only have to ensure that a small part of code is copied
- Overhead?
 - not a problem according to application level experiments
 - no benchmarks to show precisely what the overhead is

Alternative Method

- What is another method?
 - binary rewriting (code patching)
 - insert checks before every write instruction to cache
 - make sure that it is not writing into the buffer

turn on the protection after write

Prevent
DATA corruption

- What is the registry, and why is it there?
 - on reboot, need to be able to find file cache pages
 - could walk kernel data structures to find pages
 - but then would have to protect those data structures
 - where in memory, which file they belong to
 - registry needs to be stored in protected, recovered area, too- How are file cache pages restored on warm reboot?
 - kernel vm dumped to swap file
 - A user-level process sorts through mem dump to restore data

file data buffer

- Effects on file system design
 - can now turn off reliability sync. writes
 - metadata updates need to be ordered in memory
 - The same for persistent memory today
 - easy to do atomic updates (using shadowing)

- They are faced with a tough challenge: How do you convince yourself and others of RPi's reliability?
 - do not have the time to work around how it is implemented
 - have to inject faults into the system
- What kind of faults can be injected?
 - bit flips
 - kernel test, heap, and stack
 - low-level software faults
 - corrupt assignments (e.g. dead registers)
 - corrupt branches
 - delete instructions
 - high-level software faults
 - var initializers
 - mem alloc (premature free)
 - copy operations
 - off-by-one (+ to -)
 - synchronization errors

- How can files be corrupted?
 - direct
 - system accidentally writes to file data which propagates to disk
 - indirect
 - I/O procedure called with wrong parameters
- How do they detect corruption?
 - checksum memory pages
 - checksum updated after every write to buffer page
 - only detects direct corruption, why?
 - memTest
 - generates a repeatable series of I/O operations
 - it can go back and check whether data was correctly written
 - tests indirect corruption

write and get checked

→ write persist → check failure

Rio's Reliability

- They claim that Rio is even more reliable than a write-through file system. How?
 - without Rio, some faults corrupted file cache pages that were propagated to disk
 - weren't caught before making permanent
 - these faults would be caught with a protected file cache

CSE221 - Operating Systems, Yuanxuan Zhou
11/9/21

Performance

- Compared performance of Rio with file systems with various degrees of reliability.
 - memory file system
 - delay data + metadata
 - delay data, metadata delayed and logged
 - delay data, metadata sync (default UFS)
 - data,metadata sync on close
 - synch on write
 - Rio without protection
 - Rio with protection
- Bottom line
 - Rio has performance of memory file system
 - protection has 8% overhead

CSE221 - Operating Systems,
Yuanxuan Zhou
11/9/21

Discussion

- Would you use an OS that used the Rio file cache?
- Is Rio even useful without removing reliability writes?
 - added protection
 - notion of protection domains separate from address spaces is useful
 - or, separation of file buffer cache into separate protected module useful
 - Intel SGX

can be used for security protection.