High-Performance Metadata Integrity Protection in the WAFL Copy-on-Write File System

Harendra Kumar, Yuvraj Patel, Ram Kesavan, Sumith Makam

NetApp, Inc., University of Wisconsin-Madison



WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON



NetApp®
Go further, faster®

Example

Customer Data Center



"Freeing free block" panic





Recovery Run?



Support checklist

- Start recovery run (fsck like tool)
- Seek Engineering help root-cause the panic

Example

Customer

How long the recovery run will take???



Engineering

When corruption happened?

H/W fault or S/W bug?



Scribble bug or Logic bug?

India

USA





Summary

- Bugs keep coming
  - Hardware faults
  - Software bugs
- Important to protect metadata for correctness
- Need of the hour
  - Simple techniques for strong data integrity
  - No/negligible performance impact (deployable)
  - Diagnostic capability

Our Solution

- Separate solutions for separate problems
  - Deployed in production
    - Incremental checksum for scribble bugs
    - Digest-based transaction auditing for logic bugs
  - In house
    - Page-level protection for diagnostics

Key Results

- Techniques protect metadata
  - Negligible performance impact
  - More than 3x reduction in recovery runs
  - Deployed in > 250K systems worldwide
- Field data (~5 years)
  - 33 systems protected from 8 unique scribble bugs
  - 50 systems protected from 9 unique logic bugs

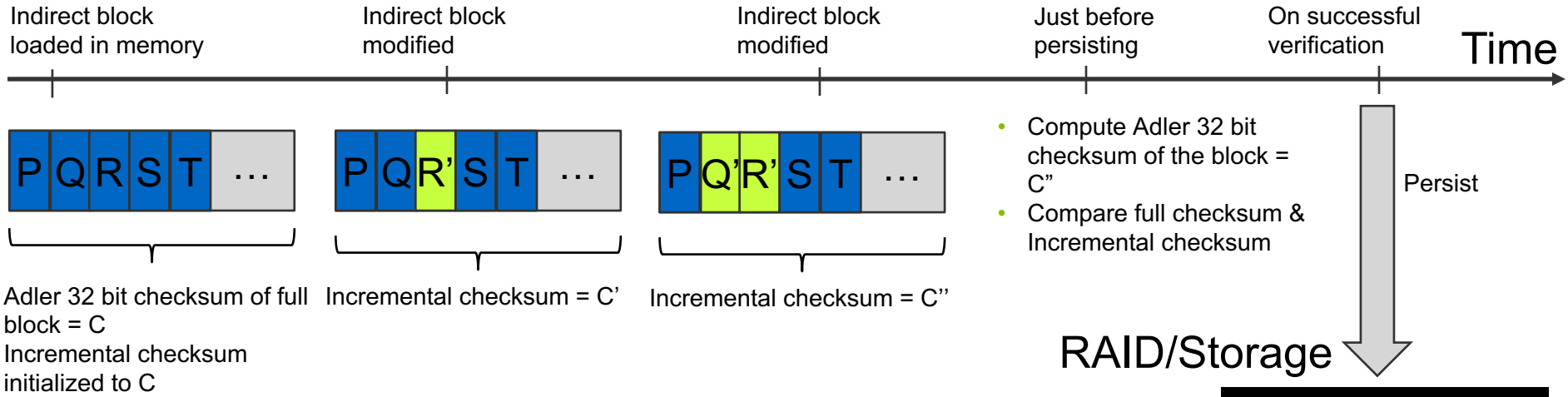
Outline

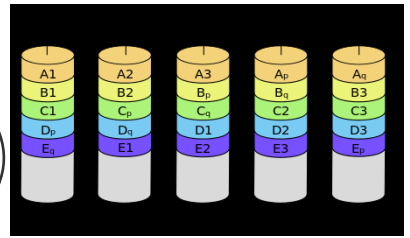
- Introduction
- Scribble protection
- Page-level protection
- Digest-based transaction audit
- Evaluation
- Conclusion

Scribble protection

- Aim: Avoid scribbles corrupting metadata
- Rolling Incremental checksum on all metadata update

Incremental checksum example



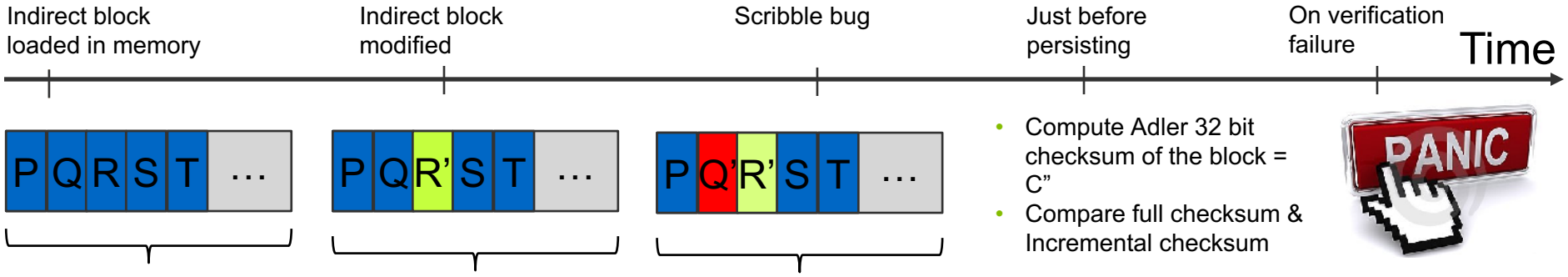


Incremental checksum computation is dependent on the amount of data modified and cache-line friendly

Metadata updates
- Small in Size
- Frequent

Concurrent incremental checksum computation possible without locks

Incremental checksum example



Without incremental checksum, this scribble bug can lead to "Freeing free block" panic

Outline

- Introduction
- Scribble protection
- Page-level protection
- Digest-based transaction audit
- Evaluation
- Conclusion

Page-level protection

- Scribble bugs only caught at the end of CP
- Difficult to root cause scribble bugs
- Page permissions + Write Protect Enable (WP) bit
  - Keep pages read-only by default
  - Flip WP bit before and after modification

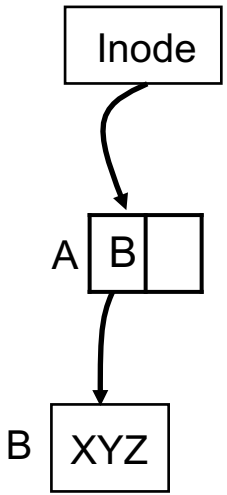
Outline

- Introduction
- Scribble protection
- Page-level protection
- Digest-based transaction audit
- Evaluation
- Conclusion

Digest-based transaction auditing

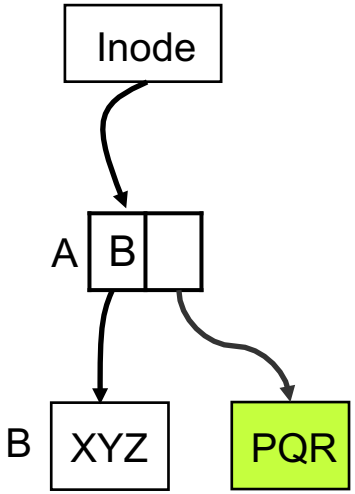
- Logic bugs and their nature
- Distributed invariants → Consistency equations
- Lightweight digest (transaction checksum)
  - Maintain different digests for different invariants

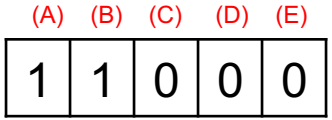
Digest-based transaction auditing



Client modifies inode A

- Adds new block

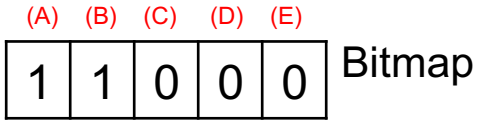


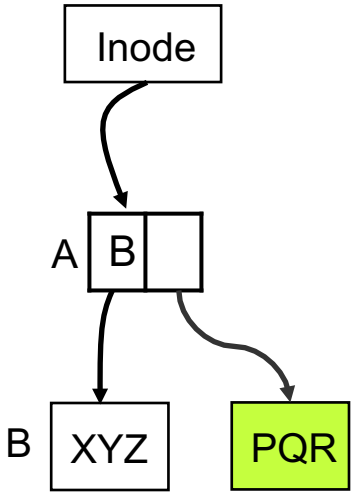


Bitmap

In-memory state of inode

Digest-based transaction auditing





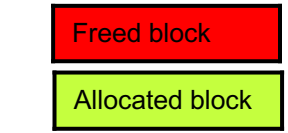
End of CP

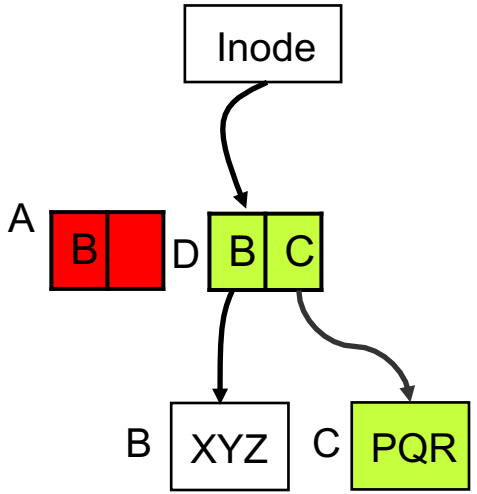
Compare digests

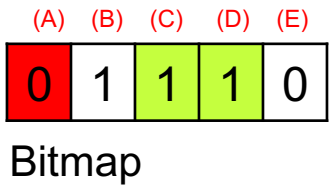
1. D1 == D3
2. D2 == D4

During CP

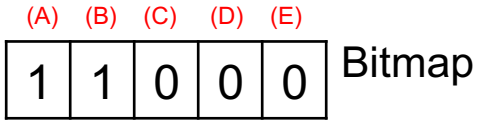
- During indirect block updates
  - Maintain blocks allocated digest D1 = C + D
  - Maintain blocks freed digest D2 = A
- During bitmap updates
  - Maintain blocks allocated digest D3 = C + D
  - Maintain blocks freed digest D4 = A

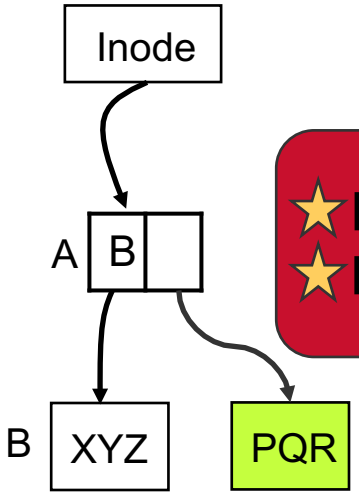






Digest-based transaction auditing





End of CP

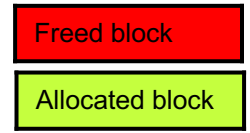
Compare digests

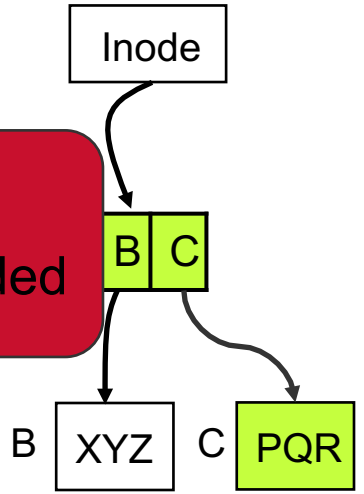
1. D1 == D3
2. D2 == D4

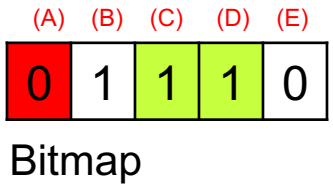
During CP

- During indirect block updates
  - Maintain blocks allocated
- During bitmap updates
  - Maintain blocks allocated digest D3 = C + D
  - Maintain blocks freed digest D4 = A

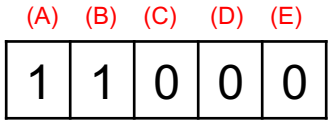
★Digests are easy to maintain
★Lightweight - Strong one to one audit avoided

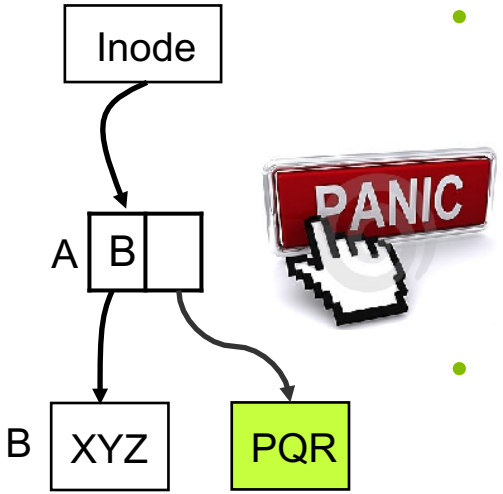






Digest-based transaction auditing





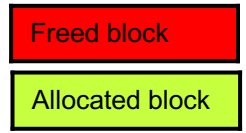
End of CP

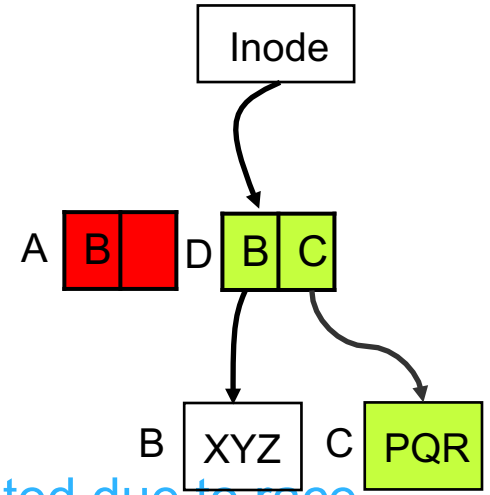
Compare digests

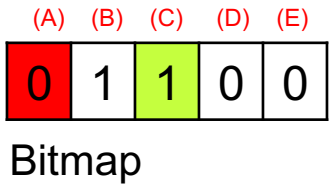
1. D1 != D3
2. D2 == D4

During CP

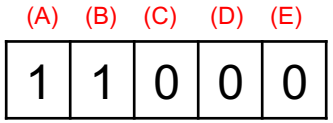
- During indirect block updates
  - Maintain blocks allocated digest D1 = C + D
  - Maintain blocks freed digest D2 = A
- During bitmap updates
  - Maintain blocks allocated digest D3 = C D not updated due to race
  - Maintain blocks freed digest D4 = A



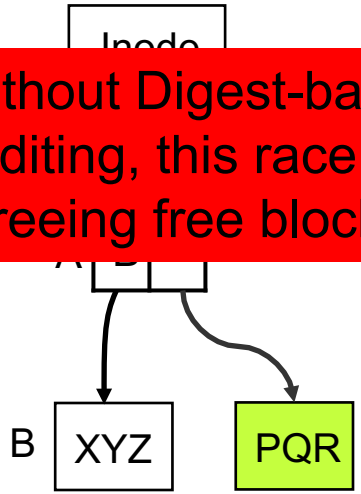




Digest-based transaction auditing



Without Digest-based transaction auditing, this race can lead to "Freeing free block" panic



During CP

- During indirect block updates
  - n blocks allocated
  -)1 = C + D
  - n blocks freed
  - digest D2 = A
- During bitmap updates
  - Maintain blocks allocated
  - digest D3 = C D not updated due to race
  - ain blocks freed
  - D4 = A

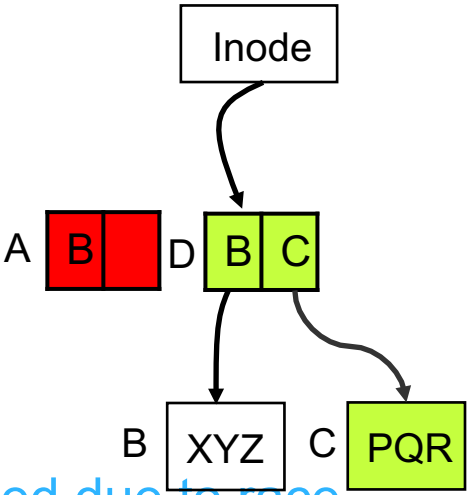


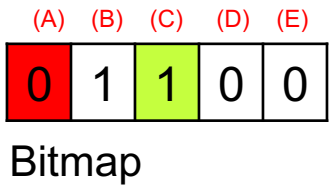
End of CP

Compare digests

1. D1 != D3
2. D2 == D4







Outline

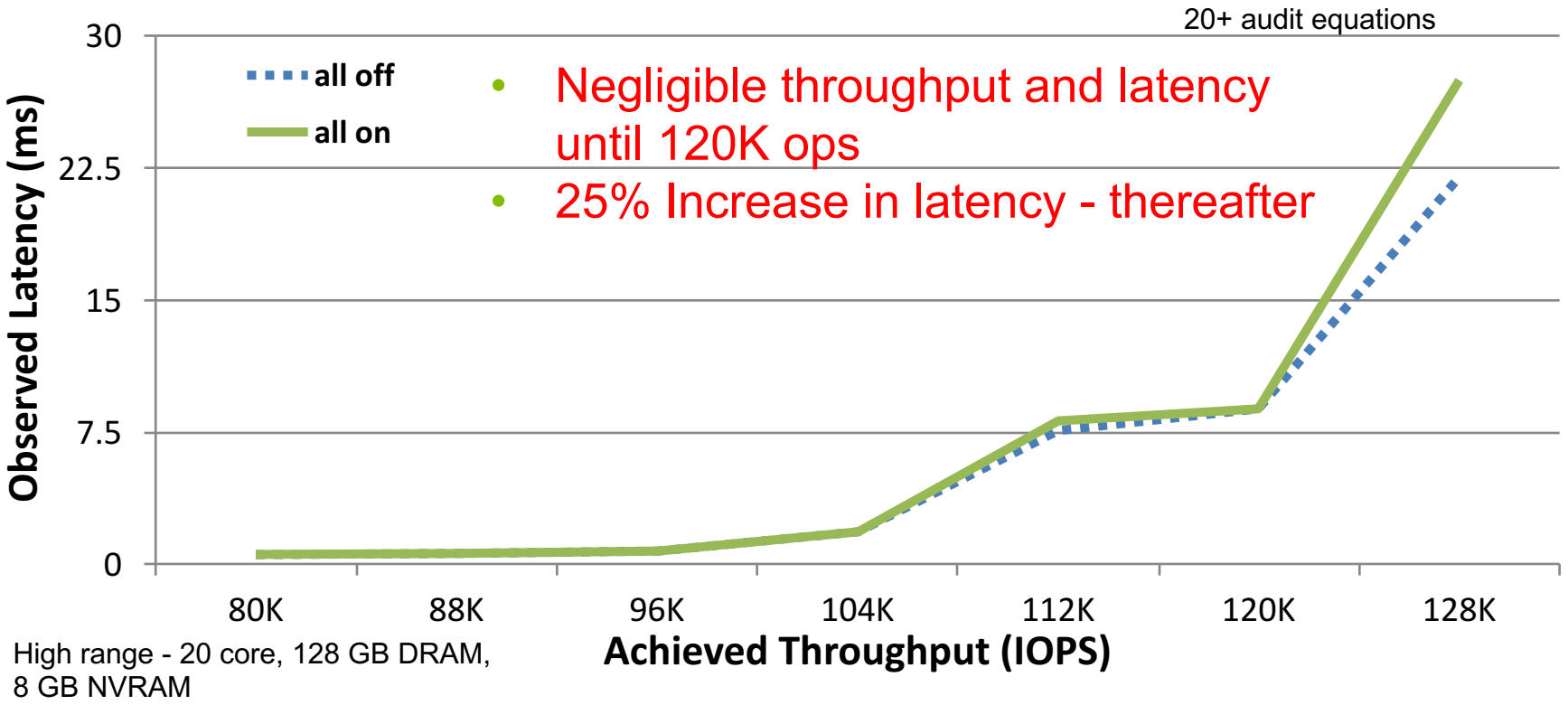
- Introduction
- Scribble protection
- Page-level protection
- Digest-based transaction audit
- Evaluation
- Conclusion

Evaluation

- Running on >250K systems for 5+ years
- Negligible regression on file server benchmarks (eg. SPEC FS)
- Heavy metadata updates by DB workloads
  - Database/OLTP benchmark (similar to SPC-1) built in-house

Performance Evaluation

Incremental checksum + Digest-based transaction auditing performance



High range - 20 core, 128 GB DRAM, 8 GB NVRAM

Performance evaluation

- Page level protection
  - 20% performance penalty
  - Used in-house (debug only kernels)
  - Only used once in field to catch a recurring scribble bug

Protection from corruption bugs

- 5 year data during in-house development
  - Unit test data hard to gather
  - 75 scribble bugs found by page-level protection
  - 32 scribble bugs found by incremental checksum
  - 23 logic bugs found by transaction auditing
- More than 3x reduction in no. of recovery runs across ONTAP 8.0 -> 8.3

Outline

- Introduction
- Scribble protection
- Page-level protection
- Digest-based transaction audit
- Evaluation
- Conclusion

Conclusion

- Introduced two techniques to enforce data integrity with minimal performance impact
- Disprove common belief - “Strong data integrity requires high performance penalty”
- End-to-end protection applicable to databases, distributed applications
- Concentrate more on innovation than worrying about data integrity

Thank you!

Questions???

