Erasure Coding in NFSv4 Back to Basics

IETF 122 NFSv4 WG Meeting

Tom Haynes - March 18, 2025

IETF 121 Feedback

- In general, more positive
- Staging Writes from file is right direction
    - Way we did it is the *worst* approach in that direction
    - Could utilize new atomic swap of blocks between two files in XFS
- Header file is too costly
    - Shrink it to a minimum
        - Want average overhead per chunk to be 8 bytes

IETF 122 Reality

Or why haven't we progressed

- Use cases are not well presented
- Proposed solutions do not scale well with
  - Use cases
  - Implementations of kernel bound servers
    - Chunks
    - Data blocks
    - Disk segments

Basic Use Case
Recovery

- “power-cut safe” during a write operation
  - power failure
  - network interruption
  - server crash
- Need to recover either:
  - Original data
  - New data

Basic Use Case

Multiple Clients

- Overwriting of data
- Need to recover either:
  - Original data
  - New data from one of the clients

Use cases

- Single Writer / Multiple Readers
  - Overwrites
- Multiple writers / but not much contention
  - Focus on block alignment
- Multiple writers / much contention
  - Of course this is HPC and must be blindingly fast

Use cases explained

- 64k clients
- Replacing Lustre
- Checkpointing

Proposed Solutions

Swap files

- Each client gets a swap file
  - Does this scale with the number of clients?
- Clone active file to swap file
- Update clone
  - How about reads of the data?
- Swap the blocks
- Many variants have been discussed

Proposed Solutions

Block Type Layout

- MDS maintains mapping of blocks to contents
- Client appends to the file
- Client tells MDS to atomically remap the file
- MDS has to recall all R/W layouts at this point
  - Grind the application to a halt
- **Probably the least likely to introduce corruption**

Proposed Solutions

Lock files

- Clients grab byte range locks when they want to write
- Removes need for distributed consensus checks
- Need to recall locks
- Need to register a lock manager for a file
    - Can spread this over the DSes
- How do we repair a file

Proposed Solutions

Distributed consensus

- Modified two touch Paxos algorithm
- Focuses heavily on the chunk
  - To be fair, many of the approaches do
- Feedback has been “No way!”

Where is the lack of scaling?

- Number of clients
- Amount of contention

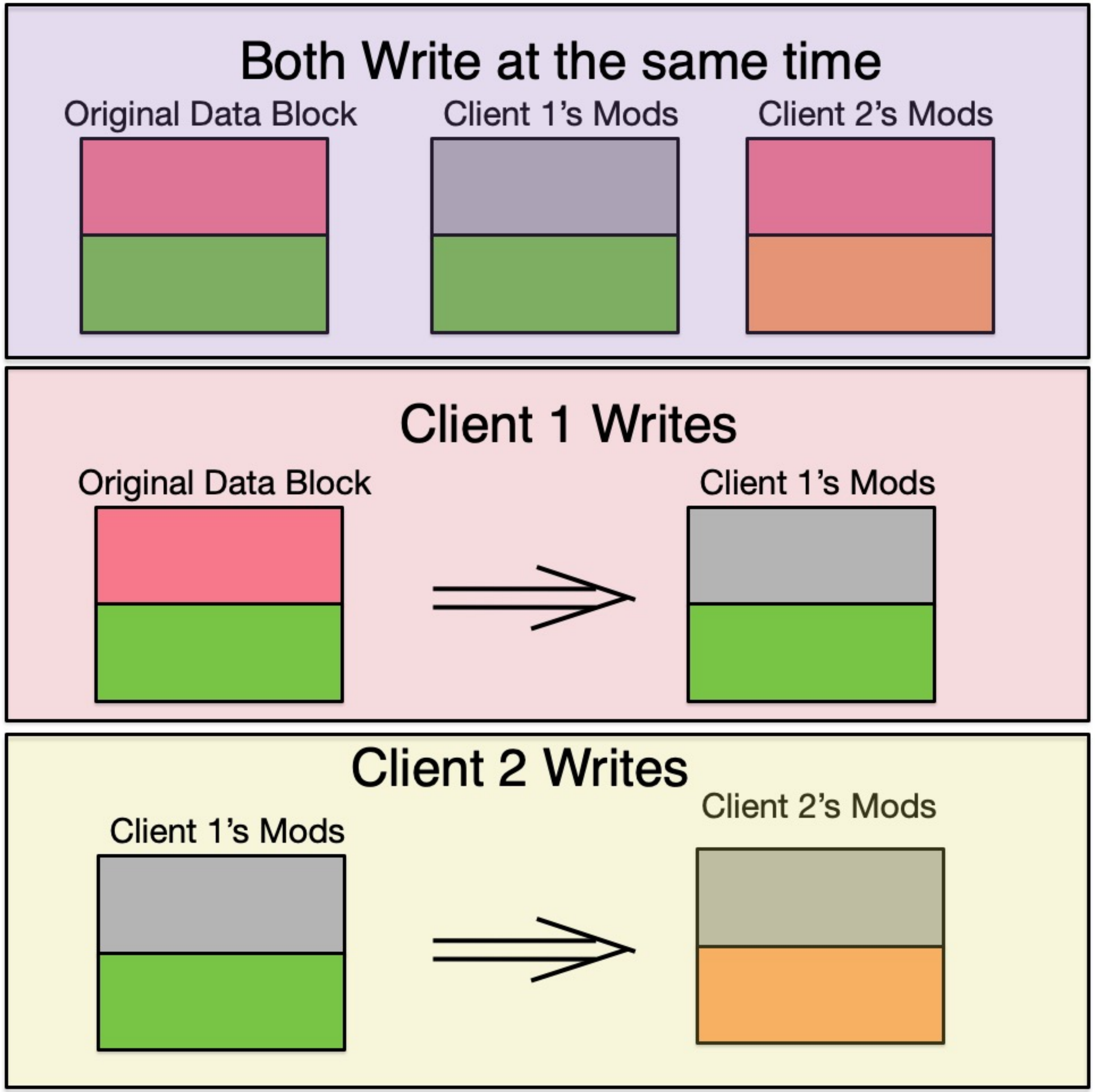
What is this kernel based limitations?

On the client

- A chunk either occupies
  - 1k
  - 4k
- Two different clients may not be conflicting in their byte range
  - Except they really are

Reminder on block contention

- Client 1 is writing start of block to middle of block - 1
- Client 2 is wring middle of block to end of block - 1
- *Perfectly safe*
- Except it is not



What is this kernel based limitations?

On the wire

- A write might be very many chunks per call
  - Can send guards per chunk, so an array of them
- This is a general problem
  - Not just kernel based

What is this kernel based limitations?

On the server

- Data blocks are 4KB, 32KB, 32MB, 64MB, 128MB
- So a chunk may be wrong, but how do you revert just that chunk in the data block?
  - File system semantics are block based
    - Not byte based
    - Not chunk based
- As we increase the block size, we increase the odds of a conflict

What is needed?

- A new paradigm where we think of chunks and not blocks.
- Changes are made in append mode to persistent storage
  - Note, not the original data file on the DS.
- Reads come first from that data file and then fall back to the data file
- As chunks are confirmed, they are written to the data file.