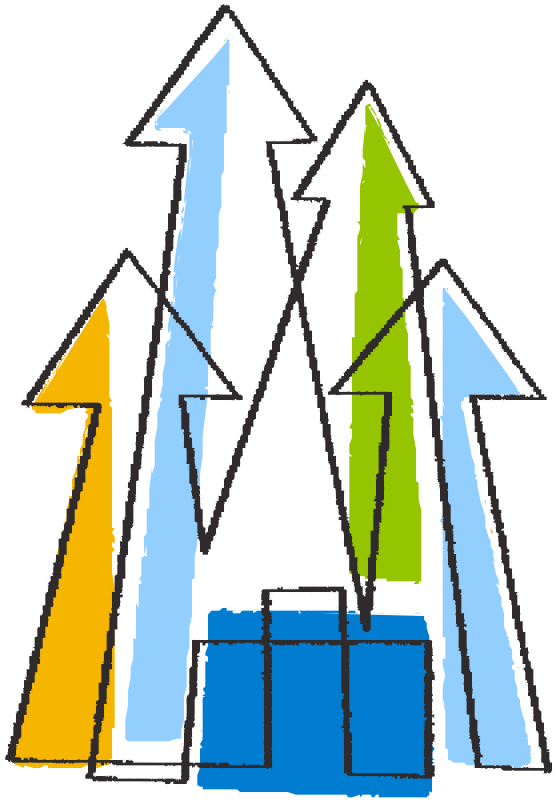
NetApp

Go further, faster®

Tradeoffs in Standards Development

Tom Haynes

2/25/14



Agenda

- Hole Punching
  - Needs asynchronous?
- Server Side Copy
  - Needs asynchronous!

Hole punching

- Done via WRITE_PLUS
  - Why not HOLE_PUNCH?
  - Why not INITIALIZE?
  - Why not VOLDEMORT?

What does it do?

- 14.7.3.2: Zero the blocks backing a particular region in the file
- wpa_hole.di_allocated == TRUE
  - Blocks will be zeroed
    - Actually write 0 to every byte of the block
- wpa_hole.di_allocated == FALSE
  - Blocks will be deallocated
- Really a hint, how does the client enforce this?

Partial blocks

- What if it is deallocate and the range is into the middle of a block?
  - Then zero that portion of the block

Benefits of WRITE_PLUS

- Do not send the bits on the wire
  - Huge savings*
- Do not store the bits on disk
  - Well, unless the server OS does not support sparse files.
  - Unloads the marshaling, sending, and unmarshaling to all be on the server

- * Even if sending 0s, might want to do a hole!

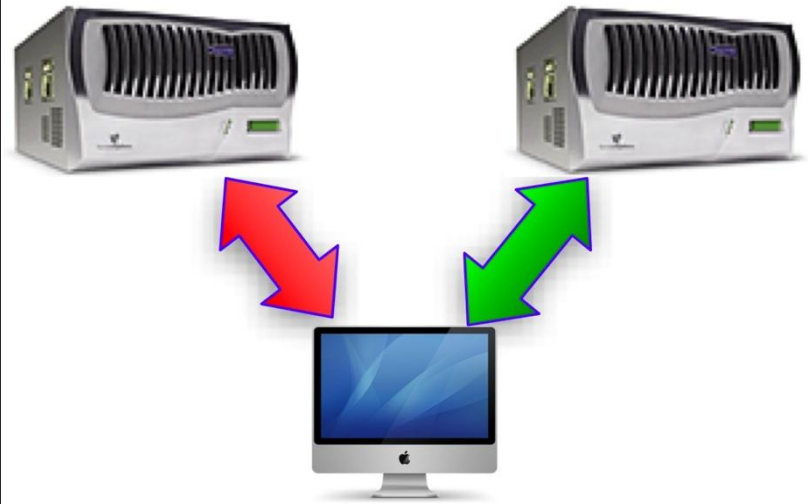
Does it need to be asynchronous?

- If true zeroing, then the server is writing the zeros
- If deallocation, should be quick
  - Mostly metadata
- What about filesystems which support dedup?

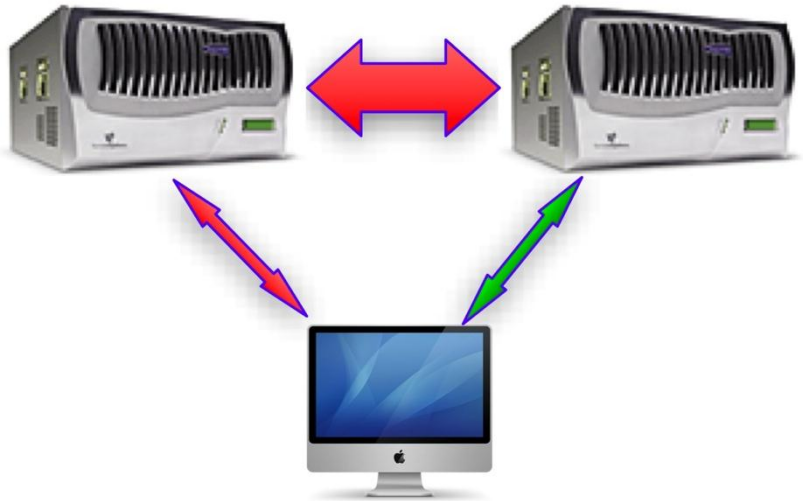
Can the server lie?

- All further reads to this region MUST return zeros until overwritten.
- If done asynchronously or if the client is waiting, then the semantics are understood
- If done as a lie, the server MUST store the WRITE_PLUS in stable storage
  - Must queue subsequent WRITES

Server Side Copy



READ from source
WRITE to destination



COPY_NOTIFY destination
COPY file from source

Vendor specific versus NFSv4.2

- Secret sauce to get bits quickly across
  - Vendor A box to Vendor A box
- NFSv4.2
  - Destination opens the file
    - *Wait, client has it open too!*
  - Destination reads the file
    - And writes locally
  - Destination closes the file

Does it need to be asynchronous?

- Yes*

- * Unless the client is willing to burn resources to wait forever

A tradeoff

- Not all servers will need to do asynchronous hole punching
- Most servers will want to do asynchronous server side copy

Voila! Reuse the asynchronous framework

- COPY_REVOKE
- COPY_ABORT
- COPY_STATUS
- OFFLOAD_REVOKE
- OFFLOAD_ABORT
- OFFLOAD_STATUS

The cost

- Are we forcing servers to support asynchronization?
  - No
  - wr_callback_id
- Are we forcing clients to support asynchronization?
  - Well, yes

What a client needs

- Clients need to be prepared for asynchronous copy offloading.
- However, for hole punching, can they indicate
  - Willing to wait until I/O is done?
  - Willing to work asynchronously?