Snap Mirror

- 2002
- NetApp

Goals? Async mirrors for disaster recovery

- Minimize data transfer
  - determine which blocks need to be transfered
- Consistent state
- Disk reads on src + writes on dest must be efficient

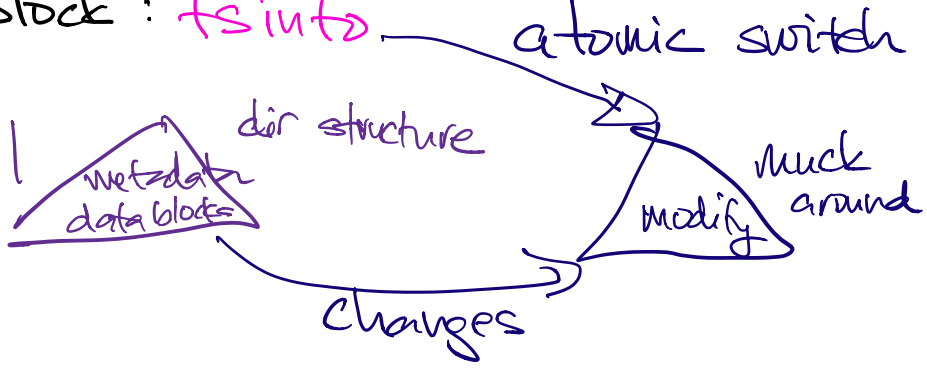
Two big ideas? (opinion)

1) Leveraging no-overwrite fs
2) Knowing which blocks are active

SnapMirror Background?

WAFL local file system
- write-anywhere, CoW

Similar to LFS; Differences?
1) writes not performed "per segment"
   individual blocks (hole plugging)
   (not as good seg. bw)
2) root block: fsinfo



Snapshots: consistent fs image @ some point in time
Need?
- fsinfo + don't reallocate blocks!

adv? (given WAFL)
- can share unmodified blocks w/ active fs + other snapshots (can't be overwritten)

data structure?
active filemap - bitmap
state of each block for this snapshot

Why useful to use bitwise OR over all snapshot's active file map?

→ block is still needed, can't be reallocated

Mirrors?

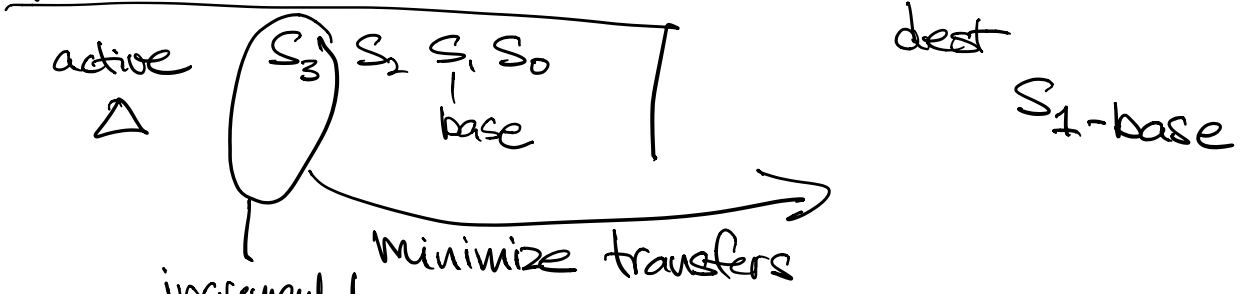
Read-only on-line replica of source file system

- Asynchronous; Impact of lag?
  - Longer ↑ → ↑ amount of data loss
  - ↓ performance impact

(don't have to update deleted items)

Where to copy?

- Copy to same logical block on dest as src
  (both RAID, don't have to be same)
- update fsinfo @ end



active map

Inc	base	
0	0	- unused - don't xfer
1	1	- unchanged, no overwrite + no reallocation!

0 1 - deleted or changed

1 0 - allocated - new or changed data

⇒ transfer

- Doesn't matter if data or metadata
- Update fsinfo @ end

Scenarios What happens if...

1) dest crashes, restarts? - just resume based on persist
- no problems (didn't update fsinfo)

3) within snapshot interval, overwrite file block C, C', C'', C'''

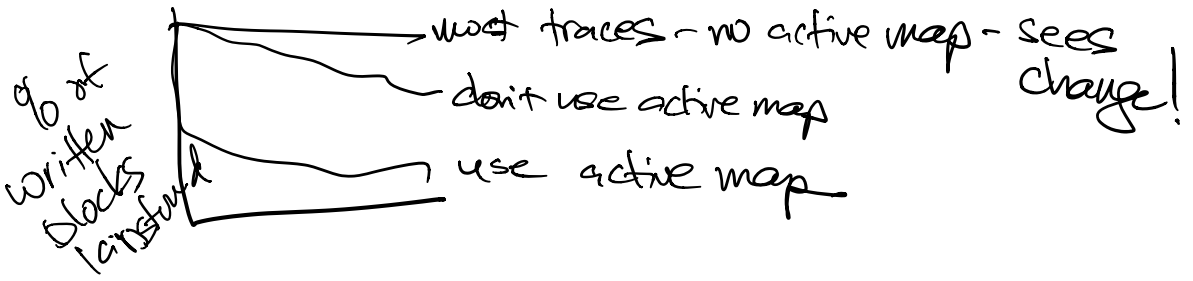
logical block array | inc active map | base active map

	inc active map	base active map
C	1→0	1
C'	0→1→0	0
C''	0→1→0	0
C'''	0→1	0

only need to transfer C'''

2) How are current updates to src fs handled? Why don't they hurt mirror?
- Only mirror snapshots not active fs

Experiment to show importance of active map? Figure 3



Summary:

- Deployed, practical system
- Leverages WAFL nicely
  - easily keep all snapshots
- Knowing which blocks are active in snapshot greatly reduces BW needed