Efficient Merge Dedup: Efficient, Failure-Consistent Deduplication on Flash
Zhuan Chen and Kai Shen (University of Rochester)

I/O Deduplication Failure-Consistency

I/O deduplication helps reduce Flash wear

- Write: ABC
- L2P mapping
- P1
- Write: ABC (ref. = 2)

Logical blocks
Physical blocks (on Flash)

Metadata maintenance
- (1) Logical-physical block mapping
- (2) Physical block fingerprints
- (3) Physical block reference counts
- Need to maintain failure-consistency of data/metadata

Existing approaches for failure-consistency
- Rely on non-volatile RAM or supercapacitors/batteries
  [Srinivasan et al. 2012; Chen et al. 2011; Gupta et al. 2011]
- Checking/repair tools [Quinlan et al. 2002]
- Redo logging (extra logging writes) [Meister et al. 2010]
- Shadowing (extra index block writes) [Tarasov et al. 2014]

Challenge: metadata & failure-consistency-induced I/O shouldn’t significantly diminish the deduplication I/O saving

Efficient Failure-Consistency

Soft updates-style I/O ordering for failure-consistency
- A failure can only leave garbage (can be reclaimed later)
  Example:
  Order 1: Map L2P → Incr. P ref., Lower-than-actual ref.
  (premature block deletion)

Cyclic dependencies
- Prevent metadata I/O merging & complicate implementation (make soft updates costly for file systems [Seltzer et al. 2000])
- A deduplication system has relatively simple semantics (compared to a file system)
- We (1) carefully design all I/O paths; (2) delay non-critical I/O (that the client completion signal does not depend on)

Anticipatory I/O delay and merging
- Short I/O delay in anticipation for near-future merging
- Reminiscent of I/O anticipation scheduling [Iyer et al. 2001]

Evaluation

- Implemented as a custom device mapper target of Linux 3.14.29
- Platforms: Atom-based tablet and Xeon-based server machine
- Mobile and server workloads with 23-73% write content duplication

Weak persistence model
- A write is returned early while corresponding physical I/O can be delayed to the next flush
  - We save 18-63% I/O writes

Strong persistence model
- A write is returned by the dedup layer after all corresponding physical I/O are returned
  - We save 15-51% I/O writes
  - Anticipatory I/O delay/merging is particularly effective