New Scrub Prefetcher

Tom Caputi
tcaputi@datto.com
Scrub and Resilver Background

- Scrubs and resilver use exactly the same code
- Scrubs happen completely in syncing context
  - After spending some time scrubbing we suspend
  - Resume next txg, reconciling any state that changed
- Scrub iteration
  - Iterate through all object sets in the pool (discovering as we go)
  - Traverse through all blocks of each object set in logical order
- Read all copies / parity of each block
  - Self healing code automatically handles fixing / reporting
Current Design

= unread
= prefetched
= scanned
Current Design

- Unread
- Prefetched
- Scanned
Current Design

= unread
= prefetched
= scanned
Current Design

- = unread
- = prefetched
- = scanned
Current Design

- White = unread
- Green = prefetched
- Red = scanned
Current Design

- White = unread
- Green = prefetched
- Red = scanned
Current Design

Legend:
- White = unread
- Green = prefetched
- Red = scanned
Current Design

Diagram:

- White = unread
- Green = prefetched
- Red = scanned
Current Design

- White = unread
- Green = prefetched
- Red = scanned
Current Design

[Diagram showing a hierarchical structure with nodes colored red (unread) and green (prefetched).]

Legend:
- White = unread
- Green = prefetched
- Red = scanned
Current Design

- = unread
- = prefetched
- = scanned
Current Design

Icons:
- White = unread
- Green = prefetched
- Red = scanned

Diagram structure:
[Diagram visualized with nodes and connections, indicating flow and status based on colors]
Current Design
Current Design: Problems

- Prefetches are held up by synchronous `arc_read()`
  - Issuing code is completely single-threaded
  - Prefetches are not issued while leaf blocks are being issued

- First prefetch below a given block is effectively useless
  - `arc_read()` called immediately after its prefetch (depth first traversal)

- Bursty IO requests
  - Scrubbing leaf -> no prefetches (most blocks in a dataset are leaves)
  - Scrubbing metadnode -> tons of prefetches
New Design (Ideal)
New Design (Ideal)
New Design (Ideal)
New Design: Additional Considerations

• Ideal prefetcher makes 2 bad assumptions
  • ARC memory available >= size of all blocks in the dataset
  • All IO can be issued in parallel (infinite disk bandwidth)

• Solution
  • Prefetch function just places IO into a priority queue
  • Prioritize blocks we will actually need first based on ZIO bookmark
  • Spin up a thread to issue prefetches from the queue and rate-limit IO
New Design

Max IOs at once = 2

Legend:
- White = unread
- Green = prefetched
- Gray = queued
New Design

Max IOs at once = 2

- = unread
- = prefetched
- = queued
New Design
New Design

Max IOs at once = 2

= unread
= prefetched
= queued
New Design

Max IOs at once = 2
New Design

Max IOs at once = 2

Legend:
- = unread
- = prefetched
- = queued
New Design

Max IOs at once = 2

= unread
= prefetched
= queued
New Design

Max IOs at once = 2
New Design

Max IOs at once = 2
New Design

Max IOs at once = 2

- unread
- prefetched
- queued

Diagram:

[Diagram showing a tree structure with nodes labeled.]
New Design: Code Changes and Applications

• ARC code adjusted so that prefetch IOs can have a read callback
  • `arc_read_done()` adjusted to provide bookmark and bp for context
    • Allows IO read callbacks to issue next prefetch easily and inexpensively

• ZFS Currently has 3 prefetching implementations (not counting zfetch)
  • `dbuf.c` (`arc_read_done()` changes help here)
  • `dmu_traverse.c`
  • `dsl_scan.c`
Questions?

Tom Caputi
tcaputi@datto.com
https://github.com/zfsonlinux/zfs/pull/6256