Introducing Fast Dedup

October 2023

Allan Jude
CTO, Klara Inc.
Agenda

1. What Makes Dedup Slow
2. DDT Data Structure
3. Log Dedup
4. Dedup Quota and ZAP Shrinking
5. Pruning and Preload
6. Benchmarks
7. Future Work
8. Q and A
What Makes Dedup Slow

Read Before Write

- Write Amplification
- DDT Sorted by Hash
- IOPS Amplification
- DDT Size
DDT Data Structure

struct ddt_key
  • [32] Checksum
  • [8] Properties (Compression, PSIZE, LSIZE)

struct ddt_entry
  • [256] 4x ddt_phys_t (DITTO, SINGLE, DOUBLE, TRIPLE)
    • [48] 3x DVAs (up to 3 copies)
    • [8] Reference Count
    • [8] Physical Birth TXG
DDT Data Structure

struct ddt_key
  • [32] Checksum
  • [8] Properties

struct ddt_entry
  • [256] 4x ddt_phys_t
    • [48] 3x DVAs
    • [8] Reference Count
    • [8] Physical Birth TXG

struct ddt_key
  • [32] Checksum
  • [8] Properties

struct ddt_entry
  • [72] 1x ddt_phys_t
    • [48] 3x DVAs
    • [8] Reference Count
    • [8] Class change timestamp
    • [8] Physical Birth TXG
Log Dedup

- Write new FDT changes to an append-only log
- Maintain these changes with in-memory AVL tree
- Once log reaches a max size or age, flush to ZAP
- Amortize cost by writing to ZAP in hash order
FDT AVL Trees

- Each TXG, move new entries to the FDT AVL
- Append changes to on-disk FDT-log object
- At pool import, read on-disk FDT-log object
- Larger logs $\rightarrow$ increased import time
- Entry lookup: Order: Dedup AVL, FDT AVL, ZAP
FDT Flushing

- Write changes back to the ZAPs
- Walk the AVL tree in hash order, do some writes
- If more to do, save a checkpoint, resume next TXG
- Checkpoint get written to the bonus buffer
- If finished, truncate the on-disk log, empty the AVL
Comparing ZAP Updates: Dedup
Comparing ZAP Updates: FDT
Other Improvements

What else can we fix about Dedup
Dedup Preload

DDT Preload

- DDT performs best when cached in the ARC
- New zpool load –t fdt command
- Load the entire FDT into the ARC
- Also can be trigger automatically at import
Dedup Quota and ZAP Shrinking

Implement a Quota on FDT Size
- Constrain growth to RAM or dedup vdev capacity
- Avoid performance cliff when DDT spills to HDDs

ZAP Shrinking
- DDT is implemented as ZAPs, shrinking required for Quota to be effective, otherwise growth never resumes
  - Will also apply to directories, as a bonus
FDT Pruning

- FDT is split into 2 ZAPs: UNIQUE and DUPLICATE
- Prune from the UNIQUE list to keep the FDT small
- Requires special handling during frees
- Purge the oldest entries that have never dedup’ed
- Use a new timestamp instead of birth time
Using Fast Dedup

- `zpool set feature@fast_dedup=enabled poolname`
- `zpool set dedup_quota=48G poolname`
- `zpool set dedup_prune_policy=60d poolname`
- `zfs_dedup_log_mem_max`
Benchmarks

How big of a difference does it make?
Testing Methodology

- 2x 512 GB SSDs, 32 GB RAM, 10 cores
- Write 8 KiB records with FIO
- Create dataset, write 8x4 GiB files (32 GiB total)
- Repeat in new datasets to increase DDT size, 10x
- After each iteration, export/import pool
- Record total writes to dedicated Dedup VDEV
Fast Dedup: Reduced Inflation

**DDT vs FDT @ Recordsize=8KB: Write Inflation**

- **8k-Legacy**
- **8k-FDT-256,copies=3**
- **8k-FDT-256,copies=1,mem_max=5%**
- **8k-FDT-256,copies=1,bs=12**
- **8k-FDT-256,copies=1,bs=17**
Fast Dedup: Reduced Inflation

DDT vs FDT @ Recordsize=8KB: Write Inflation

- 8k-FDT-256,copies=1
- 8k-FDT-256,copies=1,mem_max=5%
- 8k-FDT-256,copies=1,bs=12
- 8k-FDT-256,copies=1,bs=17
Fast Dedup: Reduced Wear

DDT vs FDT @ Recordsize=8KB: Write Inflation

- 8k-Legacy
- 8k-FDT-256,copies=3
- 8k-FDT-256,copies=1
- 8k-FDT-256,copies=1,mem_max=5%
- 8k-FDT-256,copies=1,bs=12
- 8k-FDT-256,copies=1,bs=17
Fast Dedup: Increased Performance

![Graph showing performance comparison between different record sizes and deduplication settings.]

8k-Legacy
8k-FDT-256,copies=1
8k-FDT-256,copies=1,mem_max=5%
Dedup=off

Open Source Development. Reimagined.
Future Work

What else can be improved?
Last Call for Fast Dedup Sponsorship

Expected total cost of Development and Testing will be over $200K
  o Planning for 5 people for 4+ months for development (75% complete)
  o Initial design work completed by Jude and Motin - 3 months
  o iXsystems and Klara have initiated project as Gold Sponsors

Seeking additional sponsorships for Development and Testing
  o Gold = $30K = Design reviews, contributions and joint marketing
  o Silver = $10K = Prototype access and Recognition (PR and source code)
  o Bronze = $5K = Recognition and access to Slack developer channel

Sponsorship Process
  o Fill in Sponsorship form or contact morgan@ixsystems.com
  o Pay after a full set of PRs made to OpenZFS

OpenZFS Community will contribute to final testing prior to release
Further Optimization

• By default, all DDT ZAPs are copies=3
• This was thought important for DUPLICATES ZAP
  • Loss would be catastrophic
• However, we can now prune from UNIQUE ZAP
• If we detect an unreadable part of the ZAP:
  • Leak the space to avoid data loss
  • L2+ has copies++ so damage is limited to 256 L0s
Thanks

- The entire team at Klara
- Rob Norris, Don Brady, Alex Stetsenko
- Mateusz Piotrowski, Rob Wing, Fred Weigel

- The entire OpenZFS community
- Matt Ahrens, Pawel Dawidek
- Alexander Motin, Rich Ercolani
Klara Does ZFS Development & Support

1.) OpenZFS Development Services
   klarasystems.com/zfs/zfs-custom-feature-development

2.) Klara OpenZFS Support Subscription
   klarasystems.com/support/zfs-support/
Q and A
Contact Us.

📞 +1 (213) 634-4466
✉️ contact@klarasystems.com
🐦 https://twitter.com/klarainc
Thank You