klara

ZFS Caching

Explain Like I'm 5: the ZFS ARC (Adaptive Replacement Cache)

klara

Summary & Introductions

1

Allan Jude

FreeBSD &
OpenZFS Developer

2

Klara Inc.

FreeBSD Professional
Services and Support

klara

Covered in this presentation

What is ZFS? Why all the excitement?

How does most caching work?

What is an ARC and why do I want one?

How does compression help me?

What is ZFS?

- ZFS is a filesystem with a built in volume manager
- Space from the pool is thin-provisioned to multiple filesystems or block volumes (zvols)
- All data and metadata is checksummed
- Optional transparent compression (LZ4, GZIP, ZSTD)
- Copy-on-Write with snapshots and clones
- Each filesystem is tunable with properties

Why All The Excitement?

- Copy-on-Write means snapshots are consistent and instant
- Blocks used in snapshot(s) kept when overwritten/deleted
- Snapshots allow access to filesystem at point-in-time
- No performance impact on reads/writes
- Take no additional space until blocks change
- Makes your storage ransomware-resistant
- Clones allow you to “fork” a filesystem

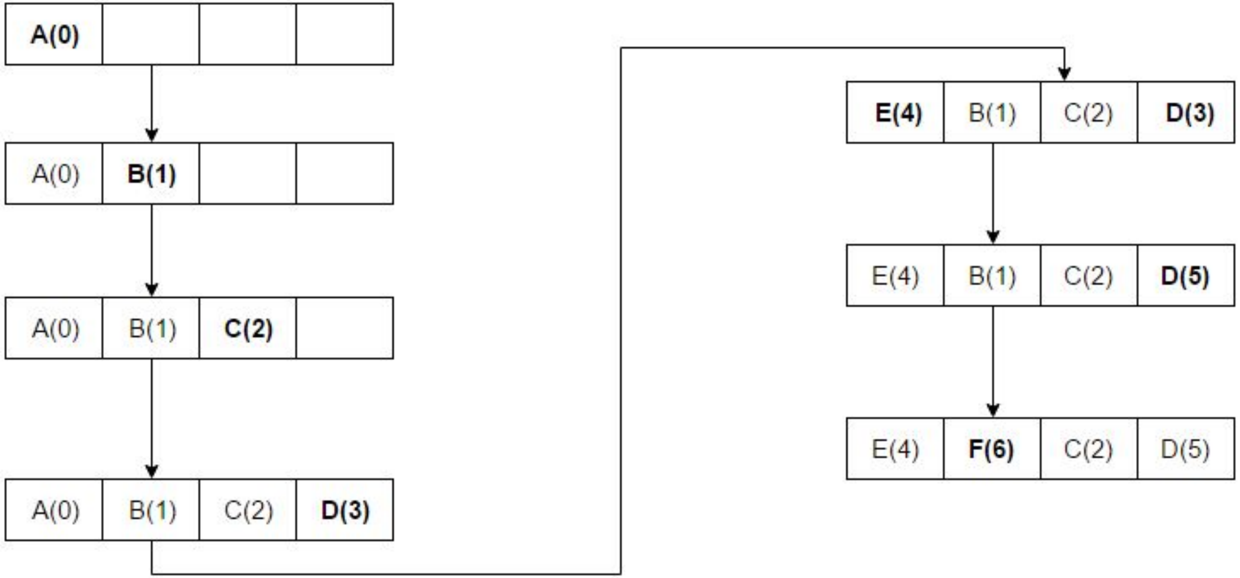
How Do Computers Work?

- Computers have multiple tiers of storage
- Each has different characteristics (speed, latency, capacity, durability)
- CPU L1 > L2 > L3 > RAM > NVDIMM > Disk Cache > Disk
- "We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible." Von-Neumann, 1946.

What Is a Cache?

- Copy of commonly and/or recently used data on faster storage
- This data can be discarded at any time, it is just a copy
- The amount of storage available in the faster tier is limited
- Faster/closer storage is a precious resource
- Need a caching algorithm to determine what to keep in cache
- Algorithm: LRU (Least Recently Used) from 1965 or earlier
- **Free RAM is wasted RAM!**

LRU: A B C D E D F



LRU: Pros and Cons

- Usually double-linked list
- Low overhead
- Locality principle: if process visits memory location, likely to revisit location or neighborhood soon

- Ignores frequency
- Does not adapt over time
- Disrupted by large scans
- Does not consider recent history more heavily

LFU: Least Frequently Used

- Same idea as LRU, except instead of keeping a timestamp we keep a hit counter. Designed in 1971
- Each time we access a page, we increase the hit counter
- When cache is full, evict the page with the lowest counter
- Unlike an LRU, scanning a database or backing up a filesystem will not thrash the cache. The infrequently accessed objects will cycle through the cache without dislodging frequently accessed pages

LFU: Pros and Cons

- Immune to large scans
- Provides best performance for hot spots
- Advanced locality principle: probability of revisiting location increased with number of visits

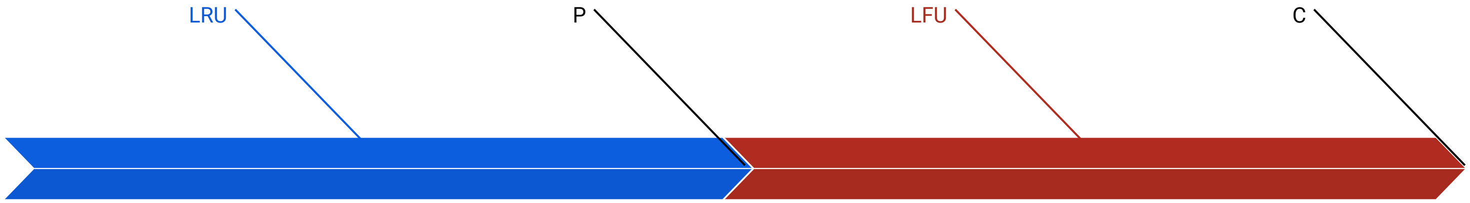
- Logarithmic complexity (slower to update)
- Does not consider recency
- Accumulates data you are no longer using

Adaptive Replacement Cache (ARC)

- Proposed in 2003 by Dr. N. Megiddo & Dr. D. Modha
- Combines the best of LRU and LFU, plus some novel tricks
- The cache size (c) is partitioned (p) into two sections
- At the start, p = ½ c, first half of the cache is LRU, second LFU
- In addition to the two caches, there is a “ghost” list for each
- Each time an item is evicted from either cache, its key (but not its data) moves to the ghost list for that cache

klara

Adaptive Replacement Cache (ARC)



LRU

Blocks that have been used recently are stored here. If a new block is read or written, it is added to the LRU. If a block is in the LRU and is accessed again, it moves to the LFU.

P

P is the adaptive part of the ARC. P starts at ½C, but can grow or shrink based on which cache is getting better hit rate, and is therefore providing the most benefit.

LFU

Blocks that are frequently used are stored here after being promoted from the LRU. If not many blocks are frequently used, P will grow, shrinking the LFU cache.

C

This is the current size of the ARC. It is usually bounded by arc_min and arc_max, but can grow and shrink in the face of memory pressure from other parts of the OS.

Adapting the ARC

- There are 4 lists, the LRU, LFU, and their respective ghost lists
- When a page is requested and is resident in the LRU or LFU: this is a hit, data is retrieved from the cache, good job!
- The page has now been accessed frequently, moves to the LFU if not already there. If it is, LFU counter is incremented
- When a page is requested and it is not resident in either cache, this is a miss, better luck next time! Welcome to LRU.

Robbing Peter to Pay Paul

- When a page is requested and it is on one of the ghosts lists: If only that particular cache had been a little bit bigger, this would have been a hit. We were so close...
- If the hit is on the LRU ghost list, increment p, making the LRU larger, and the LFU smaller. We'll get it right next time!
- If the hit is on the LFU ghost list, decrement p
- The value of p constantly changes to move towards the best mix of the LRU and LFU algorithms as your workload changes

A Whole Second (Optional) ARC

- Remember back at the beginning of this talk, tiered storage?
- The ARC is RAM, so it is fast, but you only have so much RAM
- There are devices faster than your main storage though…
- L2ARC (Level 2 ARC) uses a small amount of RAM to point to data on a high speed storage device (SSD, NVMe, NVDIMM)
- As data nears bottom of the LRU/LFU, it is copied to L2ARC
- Instead of being evicted, replaced with reference to L2ARC
- To avoid large scans, wearing out flash, fill rate is limited

Compressed ARC

- OpenZFS has transparent compression (LZ4, GZIP, ZSTD)
- This will compress data before it is written to disk, if beneficial
- In the past, blocks were decompressed and then cached
- In 2016 George Wilson changed to deferred decompression
- Each time a block is read from cache, it is decompressed again
- LZ4 decompression is 2-10 gigabytes/sec/core
- Most users saw 50-200% increase in effective cache size

Not Quite That Easy

- The ARC handles filesystem data and metadata separately
- By default, metadata is limited to 25% of the cache
- Data can not be evicted if it is in use
- The ARC is not a fixed size, it has a minimum and maximum size, and adapts between them based on memory pressure
- The original ARC algorithm assumes fixed size pages, ARC blocks can be anywhere from 512 bytes to 16 MB

ARC vs Swapcache

- There are many memory compression schemes out there
- The general idea is to compress infrequently used memory to create additional free memory (conserve space)
- Reacting when the system is under stress is less optimal
- Compressed ARC is using compression to create more cache
- Compressed ARC takes advantage of compression you were already doing anyway, decompression is faster and cheaper

Tuning the ARC

- General Purpose: Set arc_free_target = num-free-pages
  - If free memory drops below this level, the ARC will shrink
  - Combined with arc_max of 50% of memory at most
  - Greatly improves the experience on Laptop/Desktop
  - Always good to leave “some” RAM for the Browser
- Fileserver: Large ARC, increase metadata cache. L2ARC?
  - Many files = more metadata, cached = faster dir scans
- Block Storage (iSCSI): Large ARC, select correct volblocksize

klara

Tuning the ARC

- Database (A): small ARC, zfs set primary_cache=metadata, use DB buffer cache (understands which data is being used)
- Database (B): Medium ARC, small DB buffer cache,
  - high compression ratio ARC gives better hit ratio
- Hypervisor: Small-Medium ARC, reserve memory for VMs
  - Don’t make the VMs fight the ARC for memory
  - Avoid double caching on both the Host and inside the VM

QUESTIONS

???

klarasystems.com

More Resources

- Want to know more about ZFS?
  - "FreeBSD Mastery: ZFS" & "FreeBSD Mastery: Advanced ZFS"
- BSDNow.tv - Weekly podcast about the BSDs & ZFS
- @allanjude on twitter
- Want more? PapersWeLove.org "ARC after Dark":
  - https://www.slideshare.net/bcantrill/papers-we-love-arc-after-dark
  - https://www.youtube.com/watch?v=F8sZRBdmqc0
- Thanks to DrKK for extensive review feedback on these slides