CSE 486/586 Distributed Systems
Web Content Distribution---3
Case Study: Facebook f4

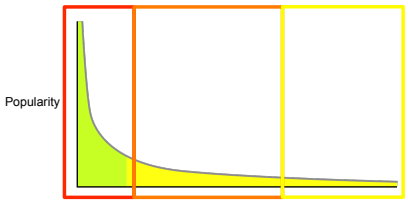
Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- Engineering principle
  - Make the common case fast, and rare cases correct
- Power law
- Haystack
  - A design for warm photos
  - Problem observed from NFS: too many disk operations
  - Mostly just one disk operation required for a photo
  - A large file used to contain many photos

f4: Breaking Down Even Further

- Hot photos: CDN
- Very warm photos: Haystack
- Warm photos: f4
- Why? Storage efficiency

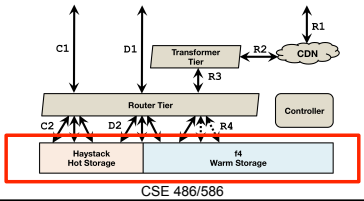


CDN / Haystack / f4

- Storage efficiency became important.
  - Static contents (photos & videos) grew quickly.
- Very warm photos: Haystack is concerned about throughput, not efficiently using storage space.
- Warm photos: Don't quite need a lot of throughput.
- Design question: Can we design a system that is more optimized for storage efficiency for warm photos?

CDN / Haystack / f4

- CDN absorbs much traffic for hot photos/videos.
- Haystack's tradeoff: good throughput, but somewhat inefficient storage space usage.
- f4's tradeoff: less throughput, but more storage efficient.
  - ~ 1 month after upload, photos/videos are moved to f4.



Why Not Just Use Haystack?

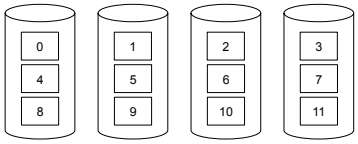
- Haystack
  - Haystack store maintains large files (many photos in one file).
  - Each file is replicated 3 times, two in a single data center, and one additional in a different data center.
- Each file is placed in RAID disks.
  - RAID: Redundant Array of Inexpensive Disks
  - RAID provides better throughput with good reliability.
  - Haystack uses RAID-6, where each file block requires 1.2X space usage.
  - With 3 replications, each file block spends 3.6X space usage to tolerate 4 disk failures in a datacenter as well as 1 datacenter failure. (Details later.)
- f4 reduces this to 2.1X space usage with the same fault-tolerance guarantee.

The Rest

- What RAID is and what it means for Haystack
  - Will talk about RAID-0, RAID-1, RAID-4, and RAID-5
  - Haystack's replication based on RAID
- How f4 uses erasure coding
  - f4 relies on erasure coding to improve on the storage efficiency.
  - f4's replication based on erasure coding
- How Haystack and f4 stack up

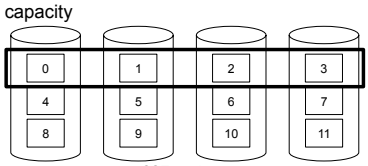
RAID

- Using multiple disks that appear as a one big disk in a single server for throughput and reliability
- Throughput
  - Multiple disks working independently & in parallel
- Reliability
  - Multiple disks redundantly storing file blocks
- Simplest? (RAID-0)



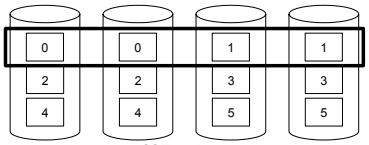
RAID-0

- More often called striping
- Better throughput
  - Multiple blocks in a single stripe can be accessed in parallel across different disks.
  - Better than a single large disk with the same size
- Reliability?
  - Not so much
- Full capacity



RAID-1

- More often called mirroring
- Throughput
  - Read from a single disk, write to two disks
- Reliability
  - 1 disk failure
- Capacity
  - Half

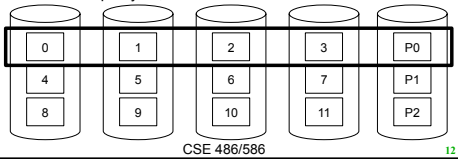


CSE 486/586 Administrivia

- PA4 due 5/6 (Friday)
- Final: Thursday, 5/12, 8am – 11am at Knox 20

RAID-4

- Striping with parity
  - Parity: conceptually, adding up all the bits
  - XOR bits, e.g., (0, 1, 1, 0) → P: 0
  - Almost the best of both striping and mirroring
- Parity enables reconstruction after failures
  - (0, 1, ✖ 0) → P: 0
- How many failures?
  - With one parity bit, one failure

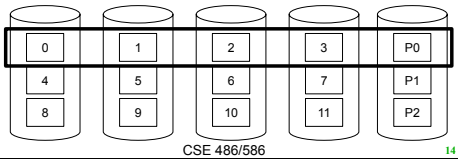


RAID-4

- Read
  - Can be done directly from a disk
- Write
  - Parity update required with a new write
  - E.g., existing (0, 0, 0, 0), P:0 & writing 1 to the first disk
  - XOR of the old bit, the new bit, and the old parity bit
  - One write == one old bit read + one old parity read + one new bit write + one parity computation + one parity bit write
- Reconstruction read
  - E.g., (0, X, 1, 0) → P: 0
  - XOR of all bits
- Write to the failed disk
  - E.g., existing (X, 0, 0, 0), P:0 & writing 1 to the first disk
  - Parity update: XOR of all existing bits and the new bit

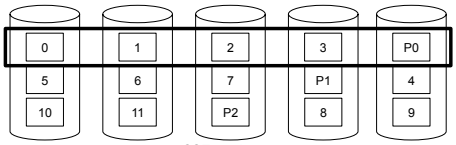
RAID-4

- Throughput
  - Similar to striping for regular ops, except parity updates
  - After a disk failure: slower for reconstruction reads and parity updates (need to read all disks)
- Reliability
  - 1 disk failure
- Capacity
  - Parity disks needed



RAID-5

- Any issue with RAID-4?
  - All writes involve the parity disk
  - Any idea to solve this?
- RAID-5
  - Rotating parity
  - Writes for different stripes involve different parity disks



Back to Haystack & f4

- Haystack uses RAID-6, which has 2 parity bits, with 12 disks.
  - Stripe: 10 data disks, 2 parity disks, failures tolerated: 2
  - (RAID-6 is much more complicated though.)
  - Each data block is replicated twice in a single datacenter, and one additional is placed in a different datacenter.
- Storage usage
  - Single block storage usage: 1.2X
  - 3 replications: 3.6X
- How to improve upon this storage usage?
  - RAID parity disks are basically using error-correcting codes
  - Other (potentially more efficient) error-correcting codes exist, e.g., Hamming codes, Reed-Solomon codes, etc.
  - f4 does not use RAID, rather handles individual disks.
  - f4 uses more efficient Reed-Solomon code.

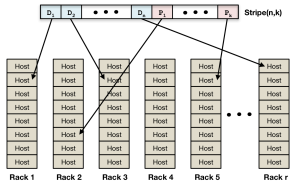
Back to Haystack & f4

- (n, k) Reed-Solomon code
  - k data blocks, f==(n-k) parity blocks, n total blocks
  - Can tolerate up to f block failures
  - Need to go through coder/decoder for read/write, which affects the throughput
  - Upon a failure, any k blocks can reconstruct the lost block.

- f4 reliability with a Reed-Solomon code
  - Disk failure/host failure
  - Rack failure
  - Datacenter failure
  - Spread blocks across racks and across data centers

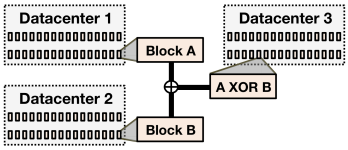
f4: Single Datacenter

- Within a single data center, (14, 10) Reed-Solomon code
  - This tolerates up to 4 block failures
  - 1.4X storage usage per block
- Distribute blocks across different racks
  - This tolerates two host/rack failures



f4: Cross-Datacenter

- Additional parity block
  - Can tolerate a single datacenter failure



- Average space usage per block: 2.1X
  - E.g., average for block A & B: (1.4*2 + 1.4)/2 = 2.1
- With 2.1X space usage,
  - 4 host/rack failures tolerated
  - 1 datacenter failure tolerated

Haystack vs. f4

- Haystack
  - Per stripe: 10 data disks, 2 parity disks, 2 failures tolerated
  - Replication degree within a datacenter: 2
  - 4 total disk failures tolerated within a datacenter
  - One additional copy in another datacenter (for tolerating one datacenter failure)
  - Storage usage: 3.6X (1.2X for each copy)
- f4
  - Per stripe: 10 data disks, 4 parity disks, 4 failures tolerated
  - Reed-Solomon code achieves replication within a datacenter
  - One additional copy XOR'ed to another datacenter, tolerating one datacenter failure
  - Storage usage: 2.1X (previous slide)

Summary

- Facebook photo storage
  - CDN
  - Haystack
  - f4
- Haystack
  - RAID-6 with 3.6X space usage
- f4
  - Reed-Solomon code
  - Block distribution across racks and datacenters
  - 2.1X space usage