CSE 486/586 Distributed Systems

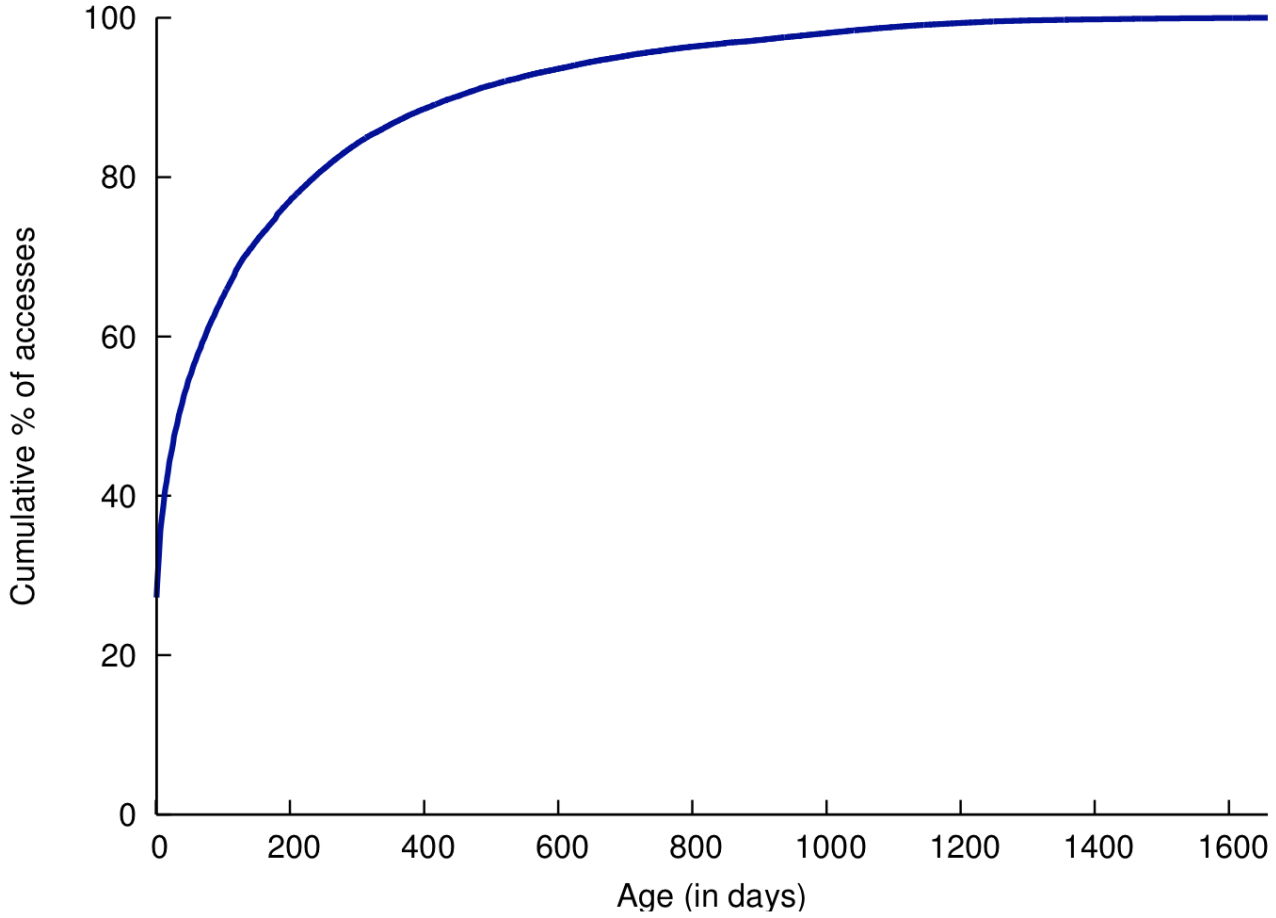
Facebook Haystack and F4

Slides by Steve Ko
Computer Sciences and Engineering
University at Buffalo

“Very warm” and “warm” Photos

- Hot photos are served by a CDN.
- Warm photo characteristics
  - Not quite so popular
  - Not entirely “cold,” *i.e.*, occasional views
  - A lot of data and views in aggregate
  - Not desirable to cache everything in CDN due to diminishing returns
- Facebook stats (in their 2010 paper)
  - 260 billion images (~20 PB)
  - 1 billion new photos per week (~60 TB)
  - One million image views per second at peak
  - Approximately 10% not served by CDN, but that’s still a lot

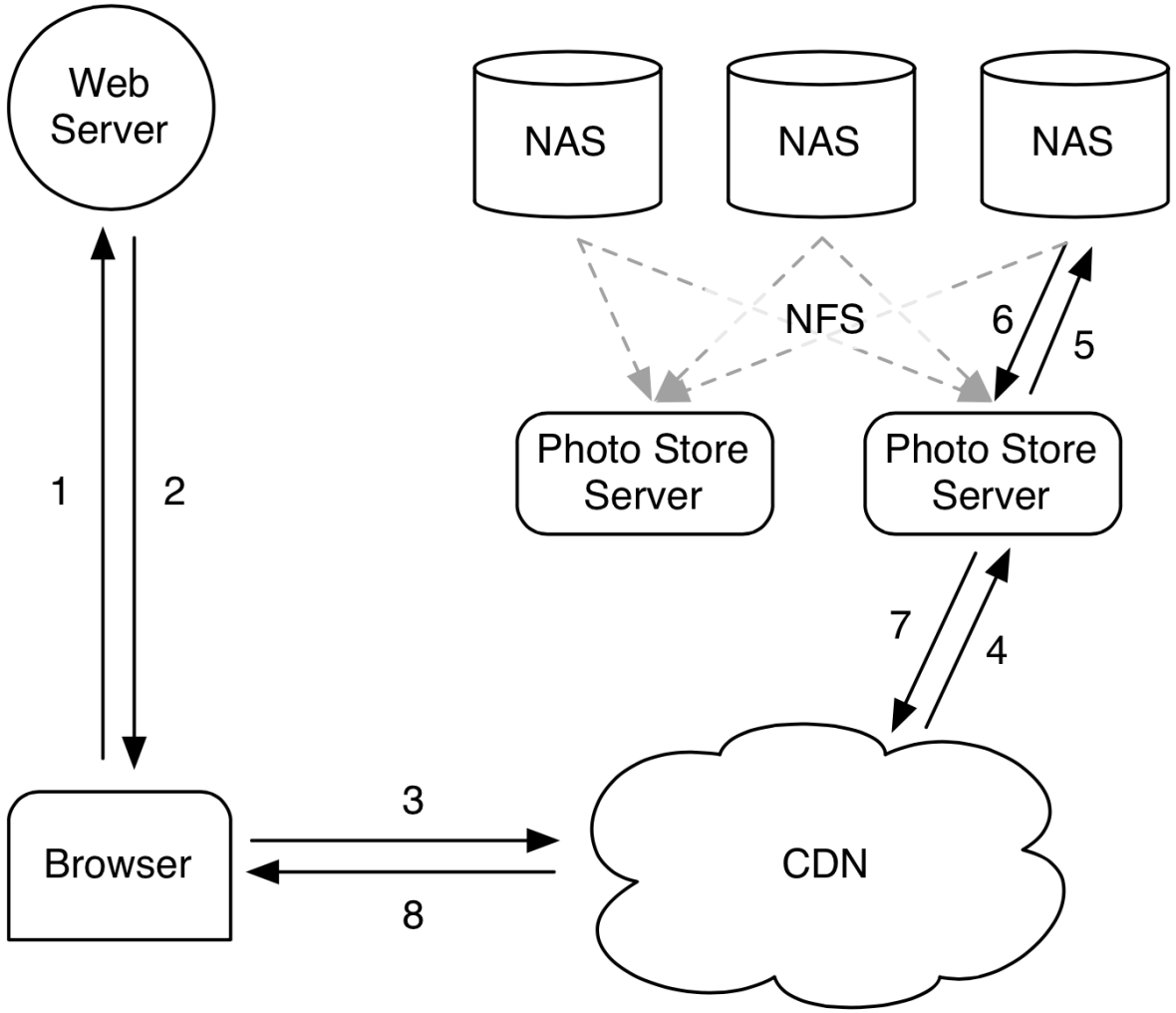
Popularity With Respect to Age



Facebook Photo Storage

- Three generations of photo storage
  - NFS-based
  - Haystack: Very warm photos
  - f4: Warm photos
- Characteristics
  - After-CDN storage
  - Each generation solves a particular problem observed from the previous generation.

1st Generation: NFS-Based



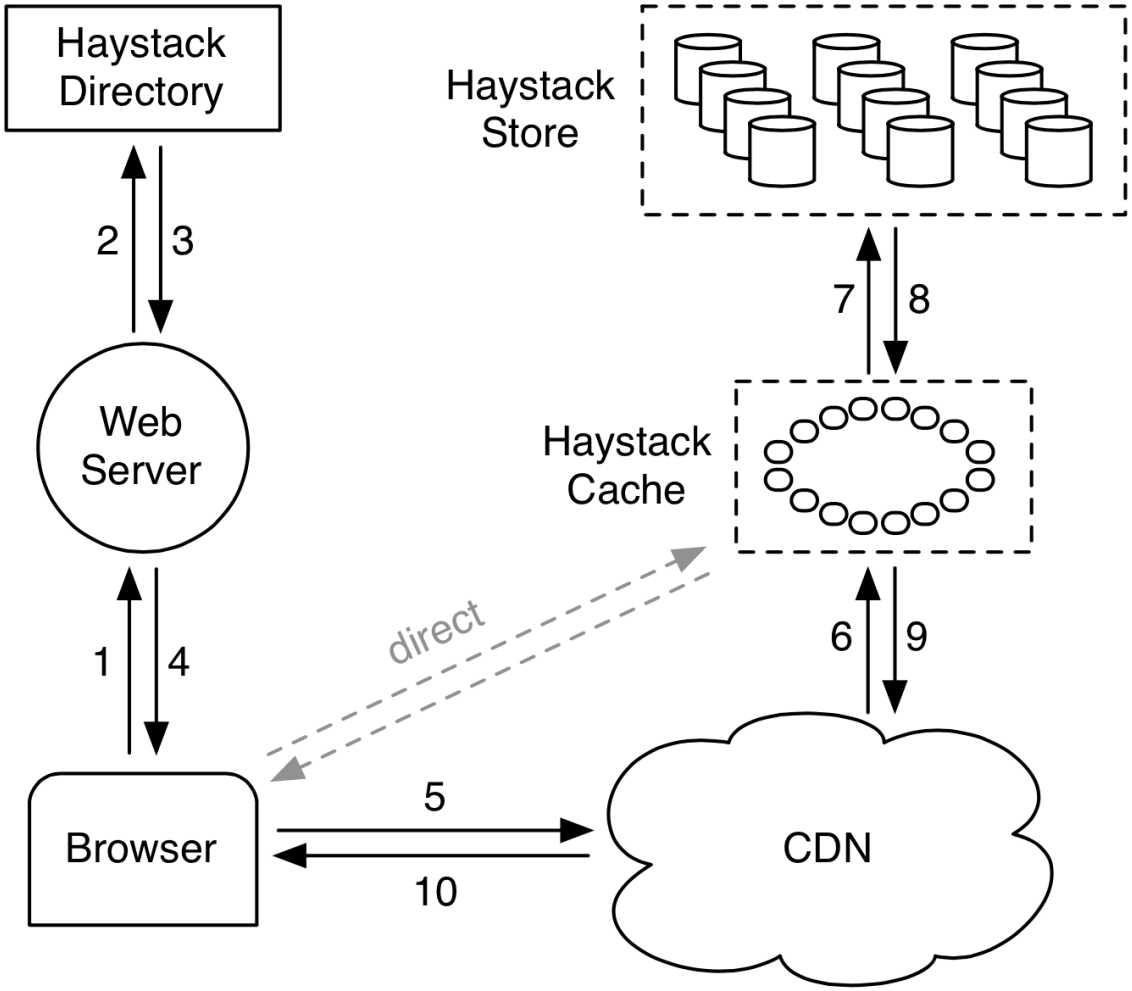
1st Generation: NFS-Based

- Each photo is a single file
- Observed problem
  - Thousands of files in each directory
  - Extremely inefficient due to meta data management
  - 10 disk operations for a single image: chained filesystem inode reads for its directory and itself & the file read
- In fact, a well-known problem with many files in a directory
  - Be aware when you do this.
  - The inode space (128 or 256 bytes) runs out.
  - A lot of operations necessary for meta data retrieval.

2nd Generation: Haystack

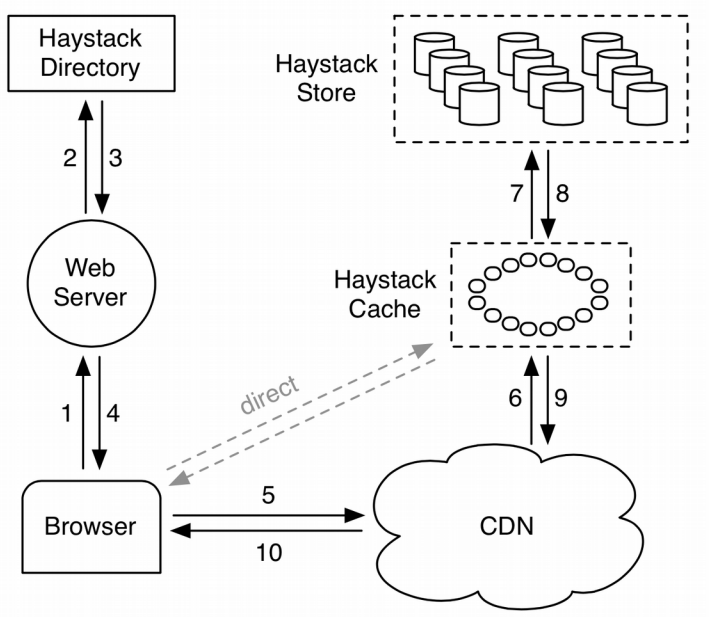
- Custom-designed photo storage
- What would you try? (Hint: too many files!)
  - Starting point: One big file with many photos
- Reduces the number of disk operations required to one
  - All meta data management done in memory
- Design focus
  - Simplicity
  - Something buildable within a few months
- Three components
  - Directory
  - Cache
  - Store

Haystack Architecture



Haystack Directory

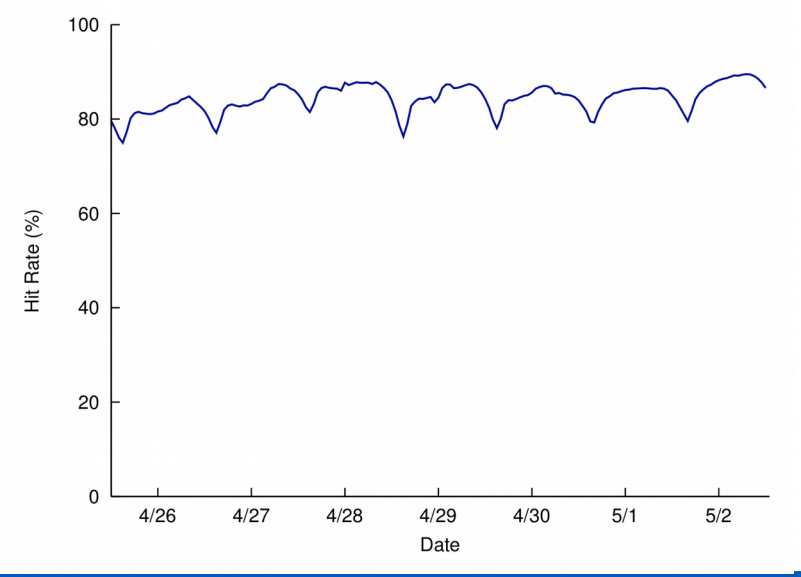
- Helps the URL construction for an image
  - http://⟨CDN⟩/⟨Cache⟩/⟨Machine id⟩/⟨Logical volume, Photo⟩
  - Staged lookup
  - CDN strips out its portion.
  - Cache strips out its portion.
  - Machine strips out its portion



- Logical & physical volumes
  - A logical volume is replicated as multiple physical volumes
  - Physical volumes are stored.
  - Each volume contains multiple photos.
  - Directory maintains this mapping

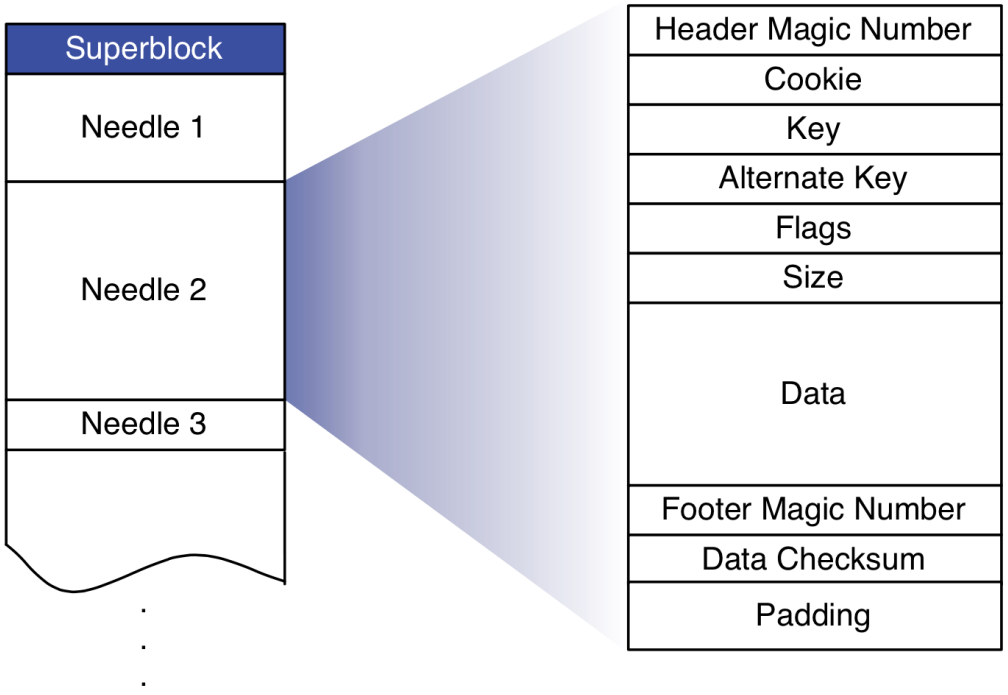
Haystack Cache

- Facebook-operated CDN using DHT
  - Photo IDs as the key
- Further removes traffic to Store
  - Mainly caches newly-uploaded photos
- High cache hit rate (due to caching new photos)



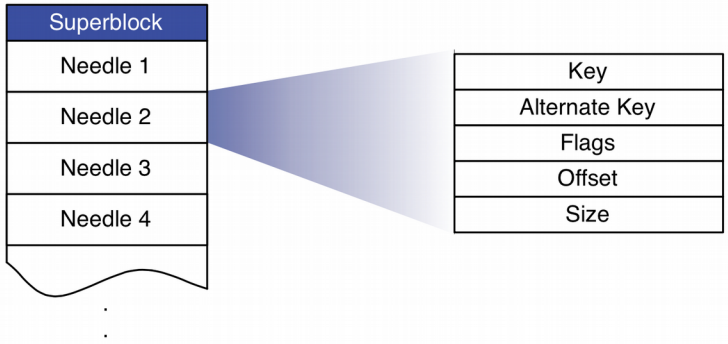
Haystack Store

- Maintains physical volumes
- One volume is a single large file (100GB) with many photos (needles)



Haystack Store

- Metadata managed in memory
  - (key, alternate key) to (flags, size, volume offset)
  - Quick lookup for both read and write
  - Disk operation only required for actual image read
- Write/delete
  - Append-only
  - Delete is marked, later garbage-collected.
- Indexing
  - For fast memory metadata construction



Daily Stats Using Haystack

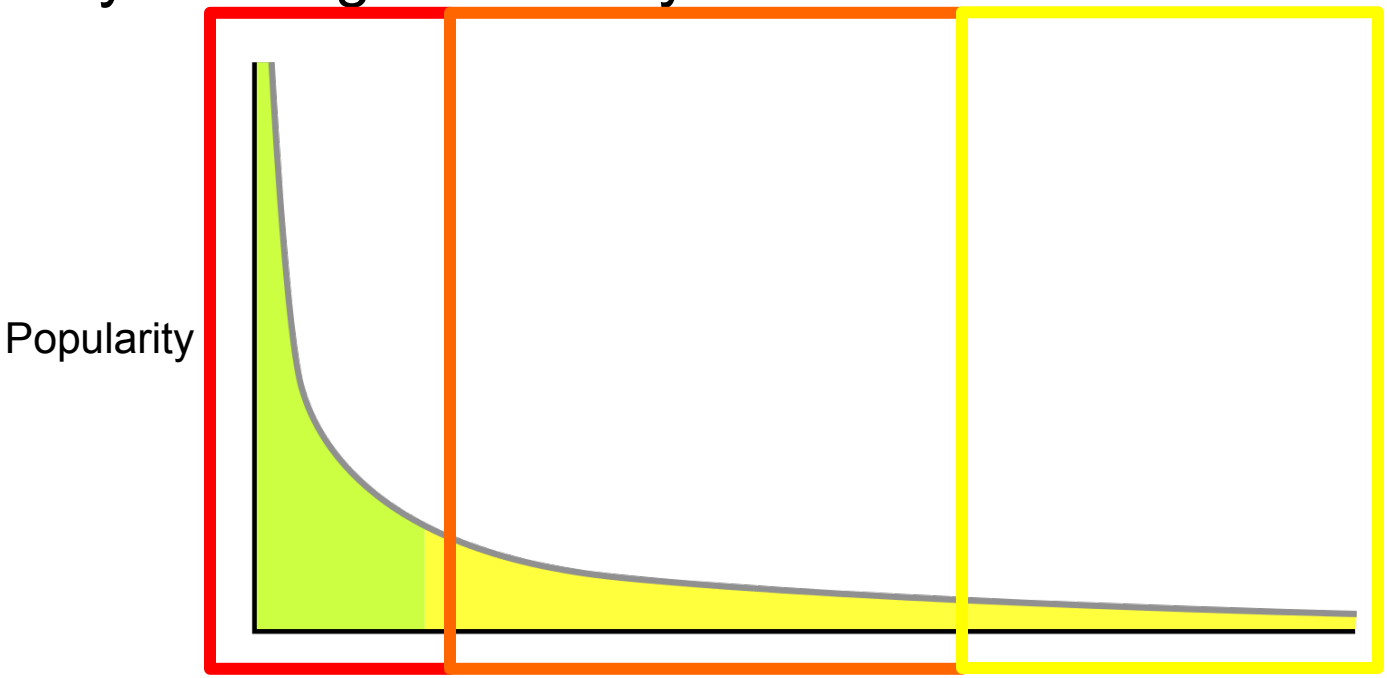
- Photos uploaded: ~120 M
- Haystack photos written: ~1.44 B
- Photos viewed: 80 – 100 B
  - Thumbnails: 10.2%
  - Small: 84.4%
  - Medium: 0.2%
  - Large: 5.2%
- Haystack photos read: 10 B

Haystack Summary

- Two different types of workload
  - Posts: read/write
  - Photos: write-once, read-many
- Photo workload
  - Zipf distribution
  - “Hot” photos can be handled by CDN
  - “Warm” photos have diminishing returns on the CDN.
- Haystack: Facebook’s 2nd generation photo storage
  - Goal: reducing disk I/O for warm photos
  - One large file with many photos
  - Metadata stored in memory
  - Internal CDN

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 need a lot of throughput.
- Design question: Can we design a system that is more optimized for storage efficiency for warm photos?

Why Not Just Use Haystack?

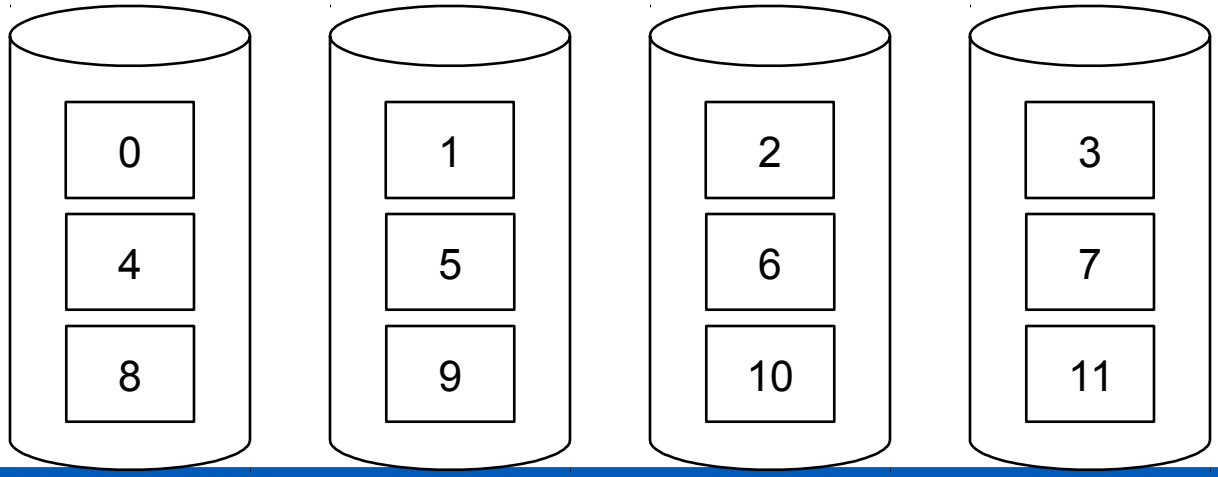
- Haystack
  - Haystack store maintains large files (many photos in one file).
  - Each file is replicated 3 times, two in a single data center, and one additional in a different data center.
- Each file is placed on RAID disks.
  - RAID: Redundant Array of Inexpensive Disks
  - RAID provides better throughput with good reliability.
  - Haystack uses RAID-6, which requires 1.2X space usage.
  - With 3 replications, each file block spends 3.6X space usage to tolerate 4 disk failures within 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
  - We will talk about RAID-0, RAID-1, RAID-4, and RAID-5
  - Haystack's replication is based on RAID
- How f4 uses erasure coding
  - f4 relies on erasure coding to improve on the storage efficiency.
  - f4's replication is based on erasure coding
- How Haystack and f4 stack up

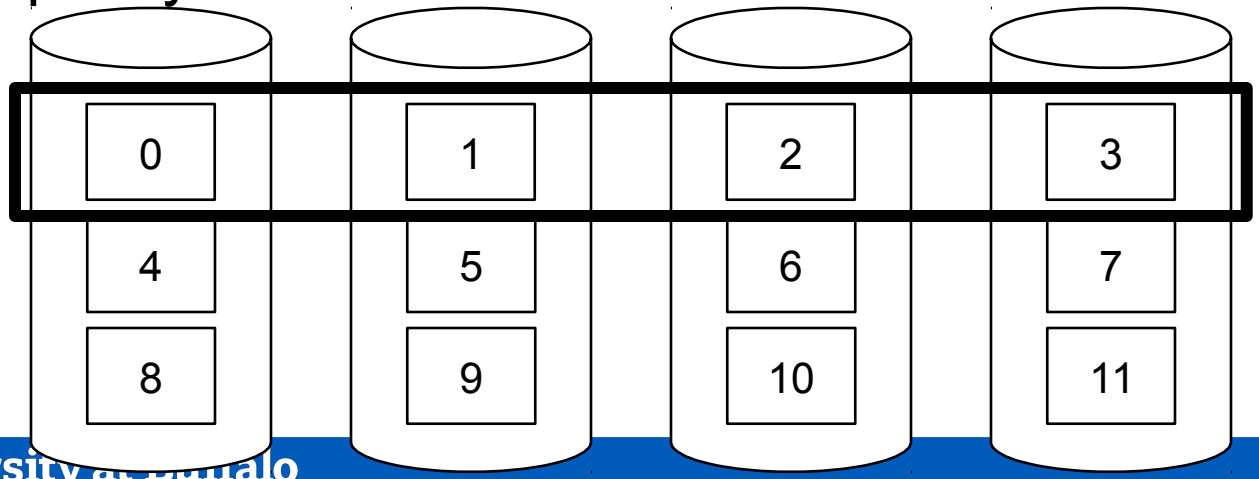
RAID

- 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)



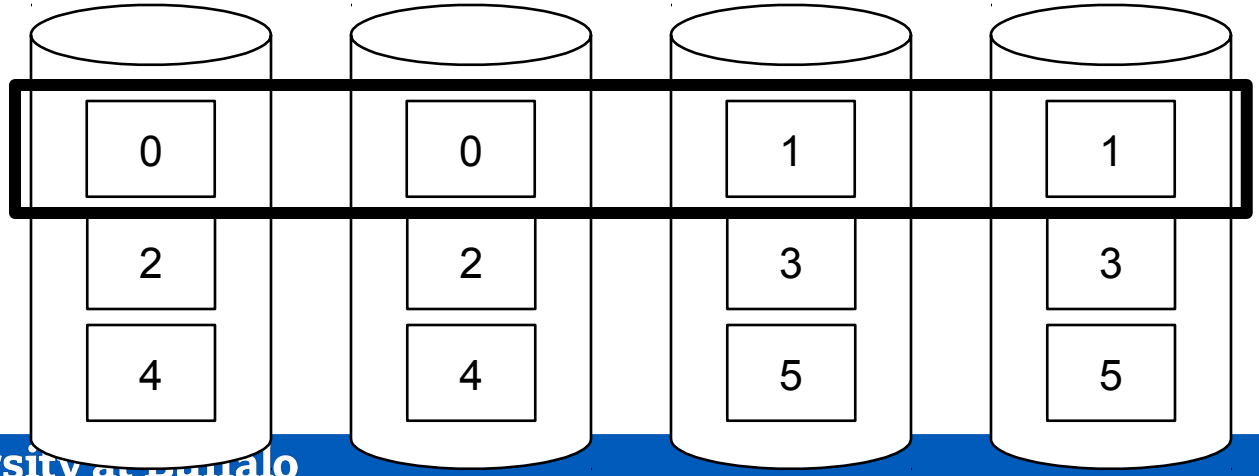
RAID-0

- More often called striping
- Provides improved 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?
  - Provides no improvement!
- Full capacity



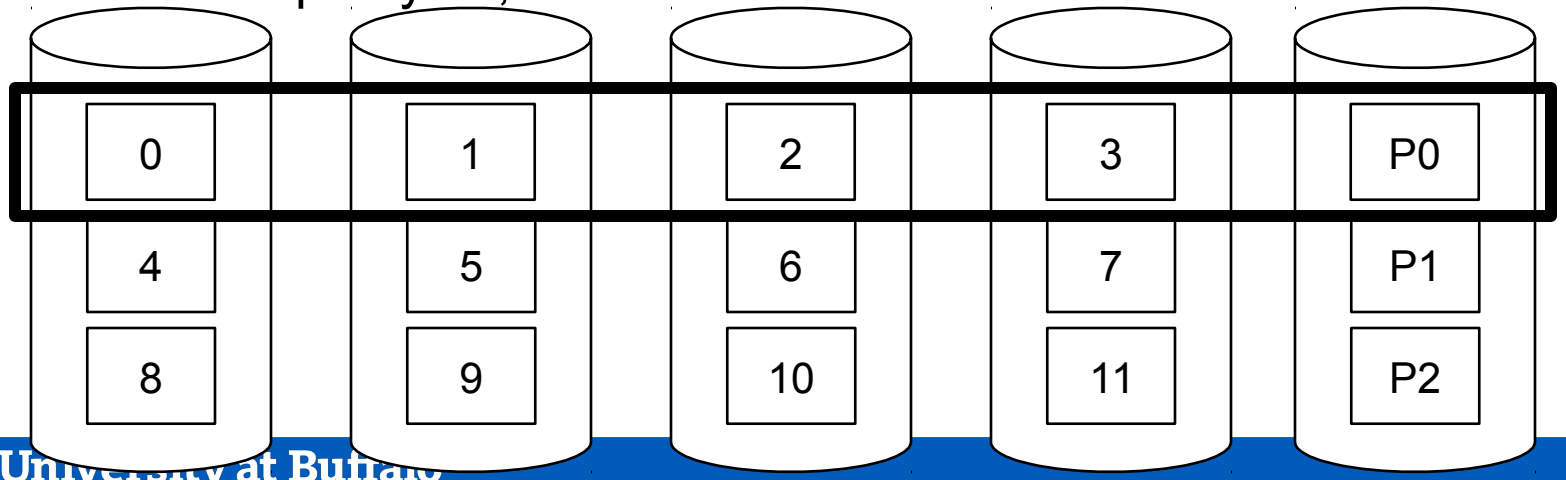
RAID-1

- More often called mirroring
- Throughput
  - Read from a single disk, write to N disks (originally 2)
- Reliability
  - N-1 disk failures
- Capacity
  - 1/N, with N mirrors



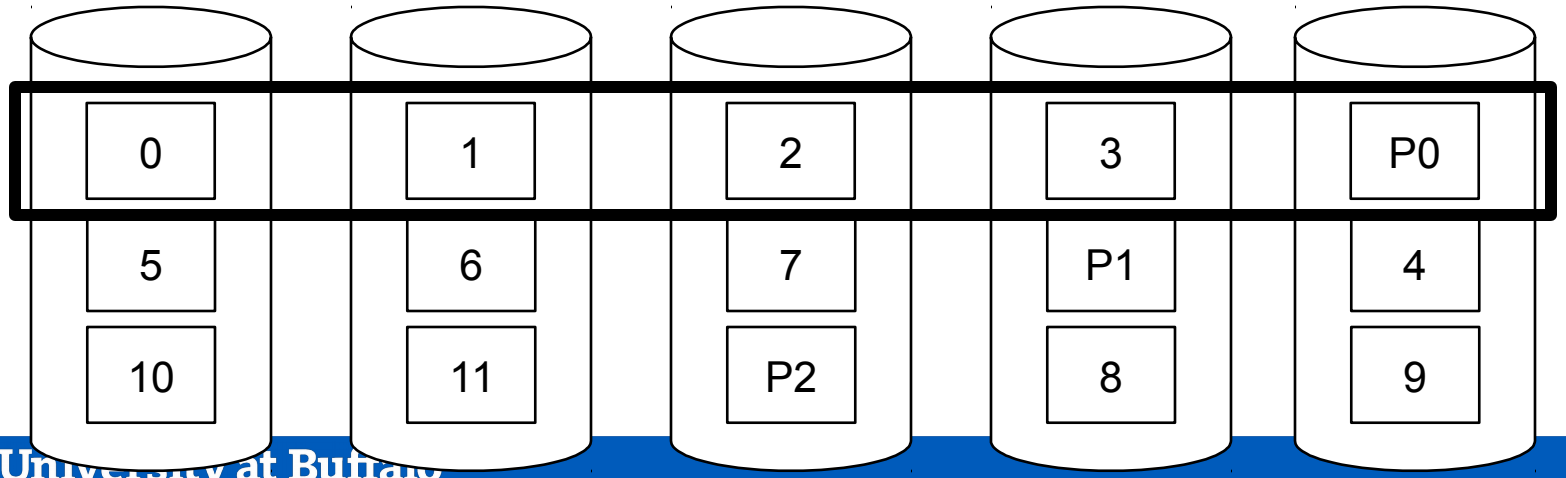
RAID-4

- 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, 1, ✖) = P: 0
- How many failures?
  - With one parity bit, one failure



RAID-5

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

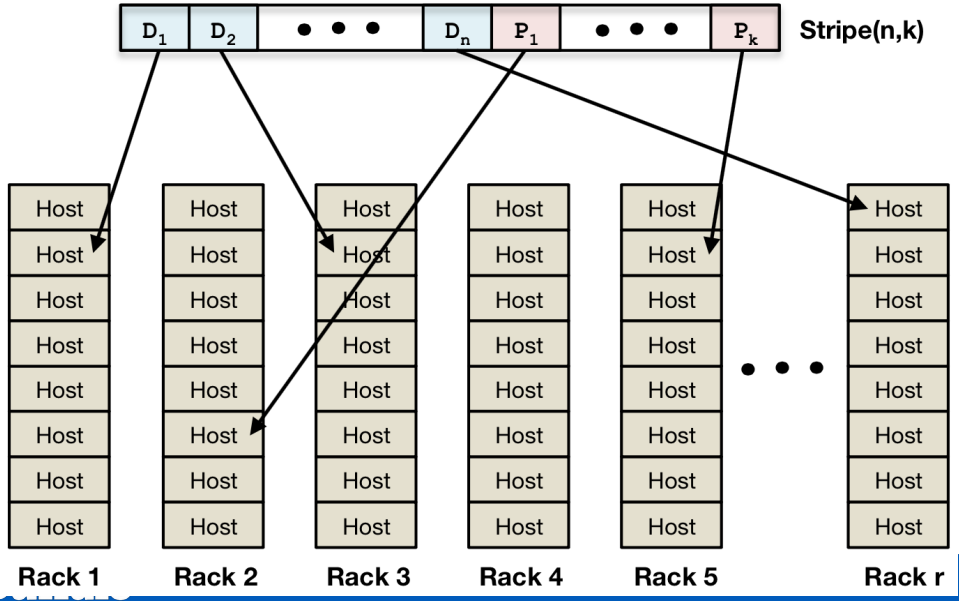


Back to Haystack & f4

- Haystack: RAID-6, which has 2 parity bits, on 12 disks.
  - Stripe: 10 data disks, 2 parity disks = failures tolerated: 2
  - (RAID-6 is much more complicated than RAID-5, 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
  - Times 3 replications: 3.6X
- How can we 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 a more efficient Reed-Solomon code.

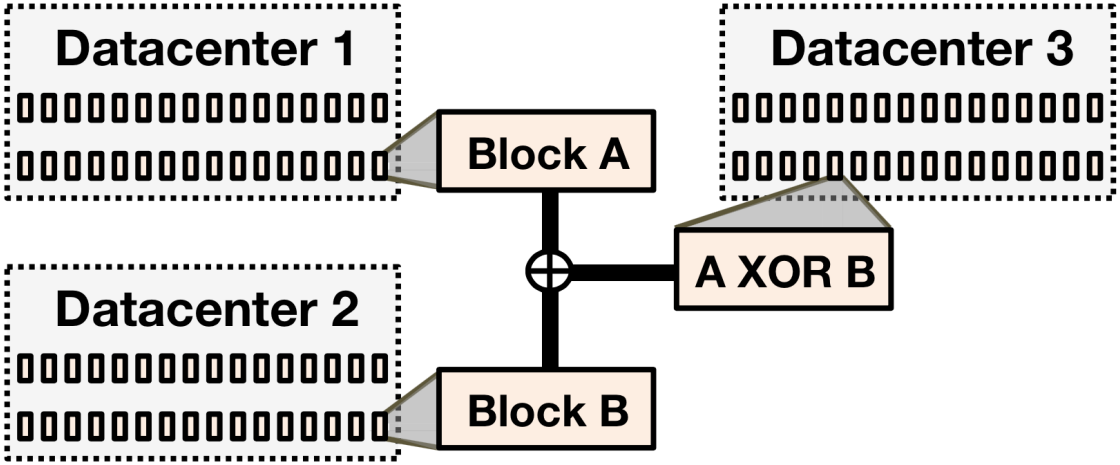
f4: Single Datacenter

- 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
  - High throughput
- f4
  - Reed-Solomon code
  - Block distribution across racks and datacenters
  - 2.1X space usage
  - Lower throughput

References

[1] Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, and Peter Vajgel. *Finding a needle in Haystack: Facebook's photo storage.* Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation. October 2010. **Required Reading.** https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Beaver.pdf

[2] Subramanian Muralidhar, Sabyasachi Roy, Cory Hill, Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar, Viswanath Sivakumar, Linpeng Tang, and Sanjeev Kumar. *f4: Facebook's Warm BLOB Storage System.* Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation. October 2014. **Required Reading.** https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-muralidhar.pdf

Acknowledgements

- These slides originally by Steve Ko, lightly modified and used with permission by Ethan Blanton.