CSE 486/586 Distributed Systems
Case Study: Facebook Haystack

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- DNS
  - Hierarchical servers
  - Root servers, top-level domain servers, authoritative servers
- CDN
  - Distributing read-only contents
  - Servers distributed world-wide
  - Server selection through DNS redirection

Understanding Your Workload

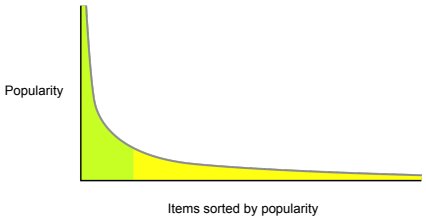
- Engineering principle
  - Make the common case fast, and rare cases correct
  - (From Patterson & Hennessy books)
  - This principle cuts through generations of systems.
- Example?
  - CPU Cache
- Knowing common cases == understanding your workload
  - E.g., read dominated? Write dominated? Mixed?

Content Distribution Workload

- What are the most frequent things you do on Facebook?
  - Read/write wall posts/comments/likes
  - View/upload photos
  - Very different in their characteristics
- Read/write wall posts/comments/likes
  - Mix of reads and writes so more care is necessary in terms of consistency
  - But small in size so probably less performance sensitive
- Photos
  - Write-once, read-many so less care is necessary in terms of consistency
  - But large in size so more performance sensitive

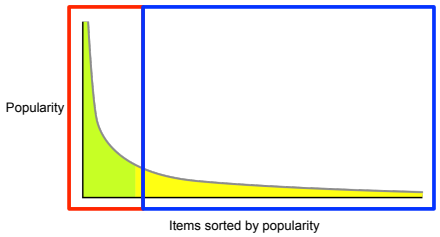
Content Distribution Problem

- Power law (Zipf distribution)
  - Models a lot of natural phenomena
  - Social graphs, media popularity, wealth distribution, etc.
  - Happens in the Web too.



Facebook's Photo Distribution Problem

- "Hot" photos
  - Popular, a lot of views
- "Warm" photos (long-tail)
  - Unpopular, but still a lot of views in aggregate



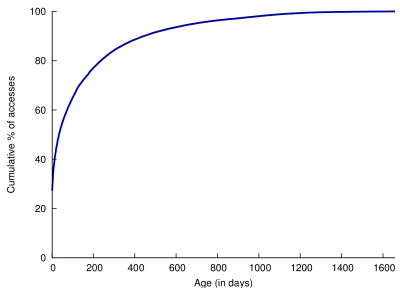
"Hot" Photos

- How would you serve these photos?
- Caching should work well.
  - Many views for popular photos
- Where should you cache?
  - Close to users
- What system gives you this ability?
  - CDN (from last lecture)

"Warm" Photo Problem

- Characteristics
  - Not so much popular
  - Not entirely "cold," i.e., occasional views
  - A lot in aggregate
  - Does not want 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 still a lot

Popularity Comes with Age



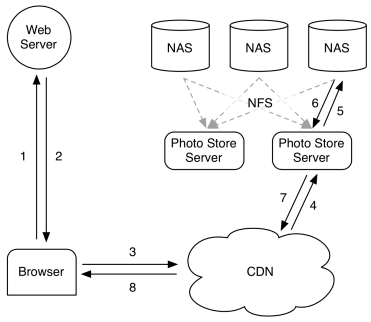
Facebook Photo Storage

- Three generations of photo storage
  - NFS-based (today)
  - Haystack (today)
  - f4 (next time)
- Characteristics
  - After-CDN storage
  - Each generation solves a particular problem observed from the previous generation.

CSE 486/586 Administrivia

- PA4 due 5/8
  - Please start now!

1st Generation: NFS-Based



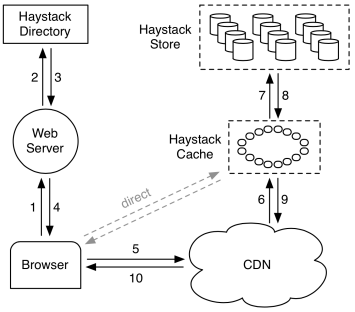
1st Generation: NFS-Based

- Each photo → 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 i-node 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.

2nd Generation: Haystack

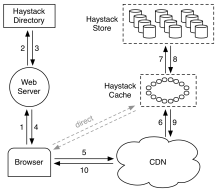
- Custom-designed photo storage
- What would you try?
  - Starting point: One big file with many photos
- Reduces the number of disk operations required to one
  - All meta data management done in memory
- Design focus
  - Simplicity
  - Something buildable within a few months
- Three components
  -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

Haystack Architecture



Haystack Directory

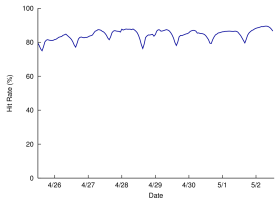
- Helps the URL construction for an image
  - http://⟨CDN⟩/⟨Cache⟩/⟨Machine id⟩/⟨Logical volume, Photo⟩
  - Staged lookup
  - CDN strips out its portion.
  - Cache strips out its portion.
  - Machine strips out its portion



- 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

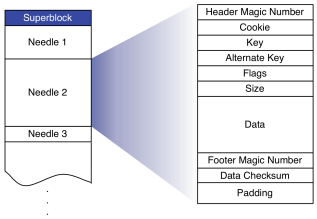
Haystack Cache

- 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)



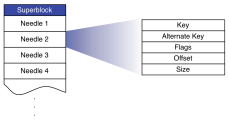
Haystack Store

- 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 with 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

Summary

- Two different types of workload for a social networking Web service
  - Posts: read/write
  - Photos: write-once, read-many
- Photo workload
  - Zipf distribution
  - "Hot" photos can be handled by CDN
  - "Warm" photos have diminishing returns.
- 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