Haystack* – The Object Store of Facebook

- Files handled by facebook include audio/video files and photographs as the platform is used primarily to share photos and audio/video content across friends.
- The file system should support quick delivery of content at a planet scale.
- One possibility is to use Content Distribution Networks (CDNs).

* Beaver et al., Finding a needle in haystack: Facebook's photo storage, USENIX OSDI 2010.

CDN Vs Facebook

- There are some fundamental differences in how CDNs work and the applications that drive Facebook.
- CDNs are geared towards optimizing access for popular (frequently used) data.
- CDNs cannot handle/optimize the case of long tail.

Other Observations

- For higher throughput, it is an advantage to keep as much metadata in memory as possible.
- This requires also minimizing the size of metadata per object.
- Identify which fields of the traditional filesystem metadata can be dropped.
- Notice that most files are not shared.
  - The owner is the only one to have the ability to update.
  - All others can only read.

The Haystack System

- Three main components
  - The Haystack Store
    - has the filesystem metadata and encapsulates the persistent storage system for photos
  - The Haystack Directory
    - maintains the logical to physical mapping along with application metadata.
  - The Haystack Cache
    - Functions as an internal CDN and serves requests for hot photos.

The Haystack Store

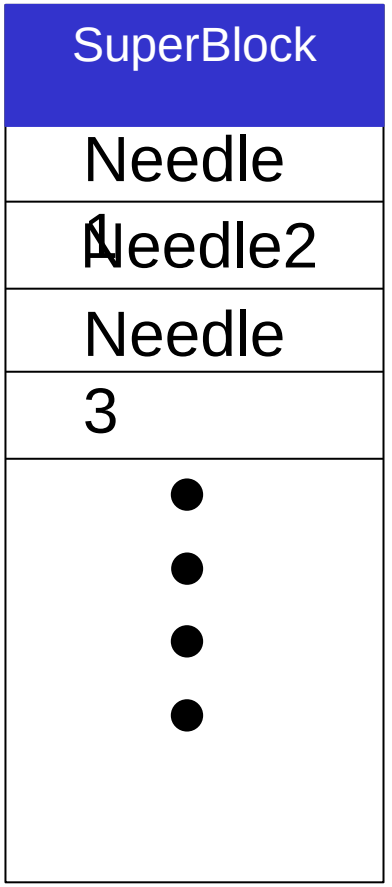
- Arranged as physical volumes with each physical volume storing millions of photos
- Multiple photos in a single file and hence the files are very large.
- Physical volumes are grouped into logical volumes.
- Haystack replicates a photo on a logical volume by writing the contents of the photo to all the physical volumes that constitute the logical volume.

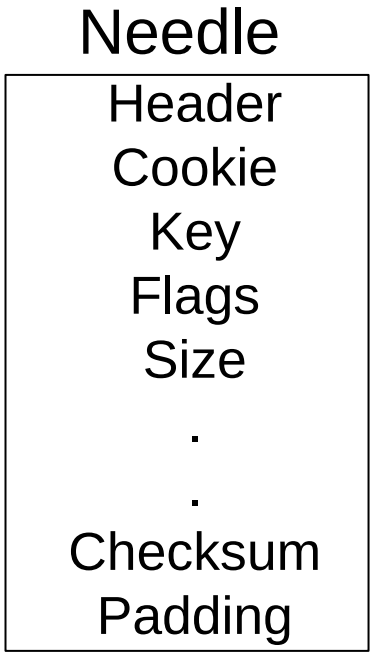
The Haystack Store

- The Haystack Store machine accesses a photo using only the id of the corresponding logical volume and the file offset at which the photo resides in the logical volume.
- A Haystack Store machine represents a physical volume as one large file.
- The file has one superblock followed by a sequence of needles.
- Each needle represents a photo stored in Haystack.

The Haystack Store

- Each Store machine is a huge physical volume.
- Typically, 100s of GBs.
- The entire volume is a single file!
- The volume is therefore viewed as a superblock with a list of needles.
- Each needle represents a photo.





The Haystack Store

- To retrieve needles quickly, each Store machine maintains an in-memory data structure for each of its volumes.
- This data structure is a hash that maps the key to other fields of the needle.
- After a crash, each Haystack Store machine reconstructs this mapping directly from the volume file before processing requests.

The Haystack Store

- Each Haystack Store machine can access a photo quickly using only the id of the corresponding logical volume and the file offset at which the photo resides.
- This knowledge is the keystone of the Haystack design.
- Retrieving the filename, offset, and size for a particular photo without needing disk operations.

The Haystack Store

- Retrieving needles quickly is an essential ability of the Haystack Store.
- Each Haystack Store machine keeps open file descriptors for each physical volume that it manages.
- Also, it stores an in-memory mapping of photo ids to the filesystem metadata (i.e., file, offset and size in bytes) to be able to retrieve that photo.

The Haystack Directory

- Four main functions
  - provides a mapping from the logical volumes to the physical volumes
  - balances the load across the logical volumes and spread reads across the physical volumes.
  - determines whether a CDN or a Cache handles the request to access a given photo
  - marks certain physical volumes as read-only once these physical volumes reach their full capacity or becuase of operational reasons
- The system adds more physical volumes as the existing volumes reach thier storage limits.
- The new volumes are marked for write. Only volumes that are marked for write can receive uploads

The Haystack Cache

- The cache keeps a mapping of the photo id to the location of the photo.
- The cache is essentially a distributed hash table.
- The cache is like an internal CDN.
- Two rules to determine if an object/photo is kept in the cache.
  - the request being served from the store comes directly from a user and not a CDN, and
  - the photo is fetched from a machine that is enabled to receive uploads.

Operations: Read/Write/Delete

- We now look at steps that Haystack uses to store a photo, serve a photo, and delete a photo.

Serving a Photo

- Consider a user requesting for a photo thorugh a browser.
- This request goes to the web server and can be served by the external CDN of the internal Haystack Cache if the request missed the CDN.
- The web server constructs a URL for the request.
- Each URL is of the form http://<CDN-Name>/<Cache-Name>/<Machine-id>/<Logical Volume, Photo>.

Serving a Photo

- The first part of the URL specifies the CDN that can possibly serve the request
- If the request is not satisfied at the specified CDN, the CDN can remove the corresponding part of the URL and forward the request to the Haystack Cache specified in the URL.
- If the Cache fails to serve the reuqest, then the Cache can remove the corresponding part of the URL and forward the request to the machine (Haystack Store) that can serve the request.
- At this level, if the photo is found in the specified volume at the given offset, it is served.

Serving a Photo

- The request to the Store is accompanied with the logical volume id, key, and the cookie fields.
- These attributes are part of the needle data fields.
- The cookie field, being a random number assigned at the time of upload and is embedded in the URL, prevents guessing attacks that can prepare valid URLs to mount denial-of-service attacks.
- The Haystack Store reads the volume from the provided offset, makes sure that the photo is not (marked) deleted, verifies the cookie, and then passes the photo to the Haystack Cache.

Storing a Photo

- When uploading a photo into Haystack web servers provide attributes such as the logical volume id, key, alternate key, cookie, and data to the Haystack Store machines.
- Each machine then synchronously appends needle images to its physical volume files and updates in-memory mappings as needed.
- Each photo is stored in four different sizes with the same key.
  - Helps address current realities such as multiple device form factors

Storing a Photo

- Operations on photos such as rotations create a new needle.
- This new needle will have the same key and alternate key as the original photo.
- There are now two possibilities.
  - The new needle is stored in the same logical volume.
    - In this case, the new needle is appended to the same physical volumes as the original needle.
    - The Haystack store uses the rule that the needle with a larger offset is considered as a later version.
  - The new needle is stored in a different logical volume
    - The Haystack Directory updates the metadata entries corresponding to the original needle.

Delete a Photo

- Deleting a photo is a logical operation.
- The Haystack Store machine that has the photo sets the delete flag in both the in-memory mapping and synchronously in the volume file.
- Since requests to retrieve photos first check the in-memory flag, such photos are never returned as the answer to the request.
- As with any logical delete models, the space held by deleted photos has to be reclaimed via a future internal operation.
- This is called compaction. Periodically,
  - each Haystack Store machine goes through the entire set of needles,
  - copies the valid needles to a new file, and
  - at the end of this process sets the in-memory structures to the new file.

Other Features

- The size of the metadata per photo stored in Haystack is about 40 Bytes.
  - Compare with ~500 Bytes possibly for the Linux xfs_inode_t data structure.
- Haystack is designed with fault-tolerance in mind.
  - Replication
  - Mechanisms such as heartbeat for identifying drives with errors

Other Popular Systems

- Google Colossus
  - More like the next version of GFS
  - Designed for much more scale.
  - Smaller block size, more metadata, metadata no longer in memory but in database!
  - Multiple Master nodes
  - Data marked as hot vs cold.
    - Hot data is used more frequently at the given time instant
      - Photos, news, etc.
    - Cold data is old data that is not used heavily.
  - Each disk to have a mix of hot and cold data.
    - Otherwise, can run into overuse/underuse.

Other Popular Systems

- The Facebook Tectonic file system
  - Generalizes from Haystack. Haystack meant mainly for blob storage.
  - Tectonic supports multiple application scenarios.