Facebook's photo storage

Tomasz Wysocki

Based on paper by Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel "Finding a needle in Haystack: Facebook's photo storage".
Problem is to

- store lot of images (260 bilion)
- be ready for new ones (1 bilion / week)
- serve them (up to 1 milion / second)
Solution must be

- high throughput
- have low latency
- fault-tolerant
- cost-effective
## Constraints

<table>
<thead>
<tr>
<th>Action</th>
<th>Normal hard drive</th>
<th>Facebook photos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>often</td>
<td>very often / never</td>
</tr>
<tr>
<td>Delete</td>
<td>often</td>
<td>rarely</td>
</tr>
<tr>
<td>Read</td>
<td>very often</td>
<td>very often</td>
</tr>
<tr>
<td>Modify</td>
<td>often</td>
<td>never*</td>
</tr>
</tbody>
</table>
Usual design
Facebook NFS approach
Haystack
Haystack Directory

1. Web Server
2. Haystack Directory
3. CDN
4. Browser
5. CDN
6. Haystack Cache
7. Haystack Store
8. Haystack Cache
9. CDN
10. Browser

direct
Haystack Cache

Diagram:
- Haystack Directory
- Web Server
- Browser
- Haystack Cache
- Haystack Store
- CDN

Arrows:
1. Direct from Web Server to CDN
2. Direct from Haystack Directory to Web Server
3. Direct from Haystack Directory to Web Server
4. Direct from Browser to Web Server
5. Direct from CDN to Browser
6. Direct from Haystack Cache to CDN
7. Direct from Haystack Store to Haystack Cache
8. Direct from Haystack Store to Haystack Cache
9. Direct from Haystack Cache to CDN
10. Direct from CDN to Browser
Haystack Store

- Haystack Directory
- Web Server
- Browser
- Haystack Store
- Haystack Cache
- CDN
Haystack Store file

Superblock

Needle 1

Needle 2

Needle 3

Header Magic Number

Cookie

Key

Alternate Key

Flags

Size

Data

Footer Magic Number

Data Checksum

Padding
Haystack Index
When file is accessed
How load is distributed (write)
Cache hit rate
Read-Only Machine
Write-Enabled Machine
Latency

![Graph showing latency over time with different machine types and dates.](image-url)
One more thing