Finding a needle in Haystack: Facebook's photo storage

Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel,
Facebook Inc.
{doug, skumar, hcli, jsobel, pv}@facebook.com

Abstract: This paper describes Haystack, an object storage system optimized for Facebook's Photos application. Facebook currently stores over 260 billion images, which translates to over 20 petabytes of data. Users upload one billion new photos (~60 terabytes) each week and Facebook serves over one million images per second at peak. Haystack provides a less expensive and higher performing solution than our previous approach, which leveraged network attached storage appliances over NFS. Our key observation is that this traditional design incurs an excessive number of disk operations because of metadata lookups. We carefully reduce this per photo metadata so that Haystack storage machines can perform all metadata lookups in main memory. This choice conserves disk operations for reading actual data and thus increases overall throughput.

1 Introduction

Sharing photos is one of Facebook's most popular features. To date, users have uploaded over 65 billion photos making Facebook the biggest photo sharing website in the world. For each uploaded photo, Facebook generates and stores four images of different sizes, which translates to over 260 billion images and more than 20 petabytes of data. Users upload one billion new photos (~60 terabytes) each week and Facebook serves over one million images per second at peak. As we expect these numbers to increase in the future, photo storage poses a significant challenge for Facebook's infrastructure.

This paper presents the design and implementation of Haystack, Facebook's photo storage system that has been in production for the past 24 months. Haystack is an object store [7, 10, 12, 13, 25, 26] that we designed for sharing photos on Facebook where data is written once, read often, never modified, and rarely deleted. We engineered our own storage system for photos because traditional filesystems perform poorly under our workload.

In our experience, we find that the disadvantages of a traditional POSIX [21] based filesystem are directories and per file metadata. For the Photos application most of this metadata, such as permissions, is unused and thereby wastes storage capacity. Yet the more significant cost is that the file's metadata must be read from disk into memory in order to find the file itself. While insignificant on a small scale, multiplied over billions of photos and petabytes of data, accessing metadata is the throughput bottleneck. We found this to be our key problem in using a network attached storage (NAS) appliance mounted over NFS. Several disk operations were necessary to read a single photo: one (or typically more) to translate the filename to an inode number, another to read the inode from disk, and a final one to read the file itself. In short, using disk IOs for metadata was the limiting factor for our read throughput. Observe that in practice this problem introduces an additional cost as we have to rely on content delivery networks (CDNs), such as Akamai [2], to serve the majority of read traffic.

Given the disadvantages of a traditional approach, we designed Haystack to achieve four main goals:

High throughput and low latency. Our photo storage systems have to keep up with the requests users make. Requests that exceed our processing capacity are either ignored, which is unacceptable for user experience, or handled by a CDN, which is expensive and reaches a point of diminishing returns. Moreover, photos should be served quickly to facilitate a good user experience. Haystack achieves high throughput and low latency by requiring at most one disk operation per read. We accomplish this by keeping all metadata in main memory, which we make practical by dramatically reducing the per photo metadata necessary to find a photo on disk.

Fault-tolerant. In large scale systems, failures happen every day. Our users rely on their photos being available and should not experience errors despite the inevitable server crashes and hard drive failures. It may happen that an entire datacenter loses power or a cross-country link is severed. Haystack replicates each photo in geographically distinct locations. If we lose a machine we introduce another one to take its place, copying data for redundancy as necessary.

Cost-effective. Haystack performs better and is less