facebook
A Day in the Life of a Facebook Photo

Brian Cabral
Director of Engineering
968 million daily users
2 Billion photos shared daily
100s of different devices

1,000s of locales/connection types
Massive distributed system

Requiring on-the-fly resizing and transcoding
Basic architecture

Photo Upload

HTTP post

PHP Front-end

Resize & Transcode

HTTP post

Everstore
Resizing

On upload we target 960 or 2048
  • function of input size & expected output resolutions

Custom resizing reconstruction filter
  • Type of separable Lanzcos filter
  • Designed for speed & quality

Resizing  re-transcode
Transcoding

We use PJPEG with custom tables

Extensive internal analysis showed that for most images PJPEG was hard to beat
Basic architecture

Photo Download

Origin Cache → Edge

HTTP get

Thrift

Everstore

Resize & Transcode

FCDN

Apple

Android

Mozilla

Firefox
CDN Caching

Facebook Data Centers

Edge Network dozens of PoPs around the world

Haystack: Distributed File Store

Proxygen: Layer 7 Balancer

BigCache: RAM + Flash Cache

Proxygen: Layer 7 Balancer

BigCache: RAM + Flash Cache

Akamai
System design & development

Continuous design & development

Controlled rollouts

Extensive instrumentation

- Big spec, design, ... waterfall
- Extensive QA testing
Open questions

Can we do better than PJEG?

Can we eliminate intermediate processing?

Is there a way to arbitrarily deliver different sizes and quality?

Can we use image specific caching schemes?

Can we use ML to dynamically load adapt?

How do GPUs and CPUs aggregate in the “large?”