f4: Facebook's Warm BLOB Storage System

Vaijayanth Raghavan
(vraghvn2@Illinois.edu)

What is a Blob?

???

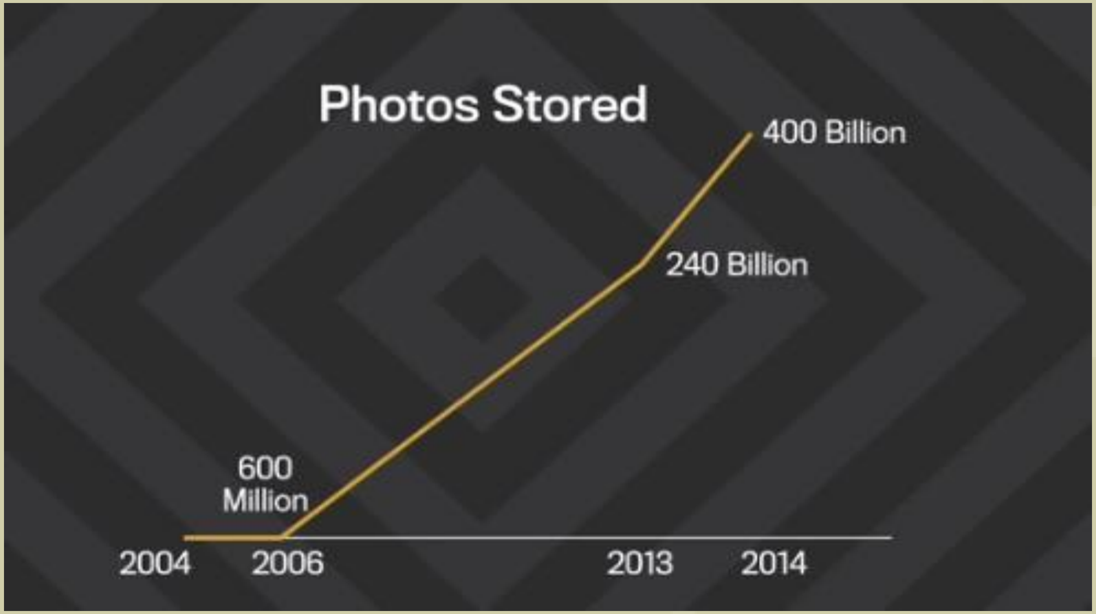
???

What is a Blob?

- Binary Large Objects
- Immutable binary data
- Created once, read many times, never modified and sometimes deleted
- Includes photos ,videos ,documents (visible to the user) and others like heap dumps, source code (internal to FB)
- As of Feb 2014, 400 billion photos

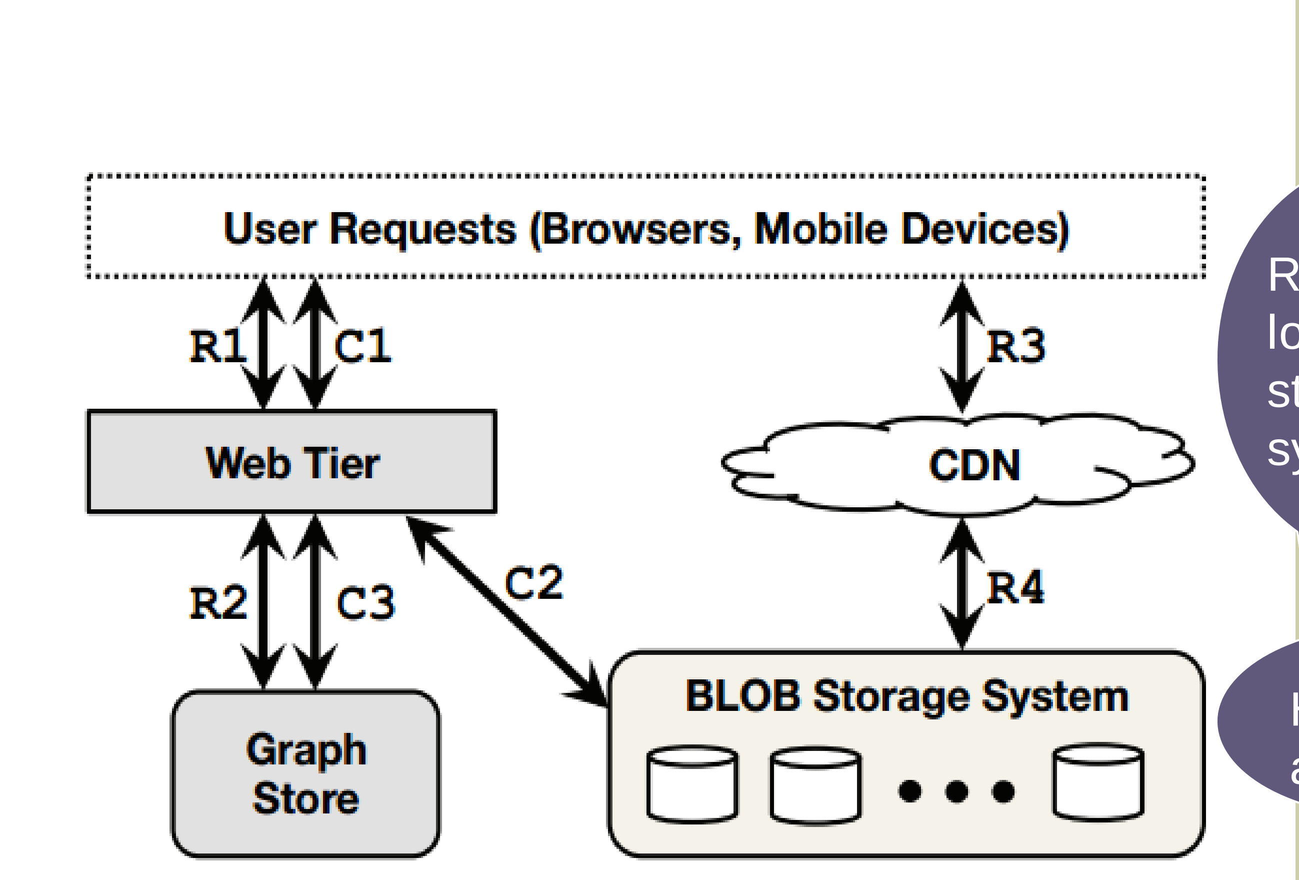
  (huge storage footprint)

FB photo growth



400 billion and growing !!!

Overall architecture

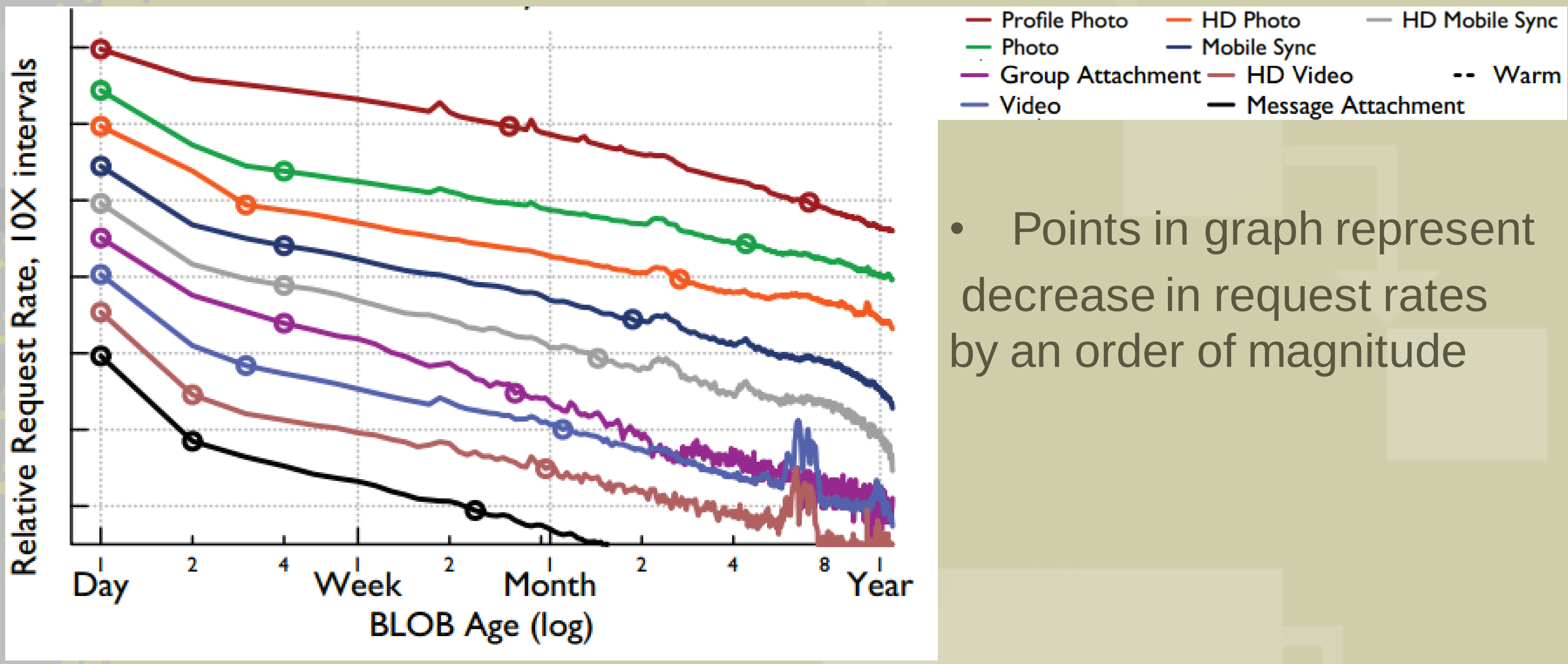


Reduces load on storage system

Haystack and f4

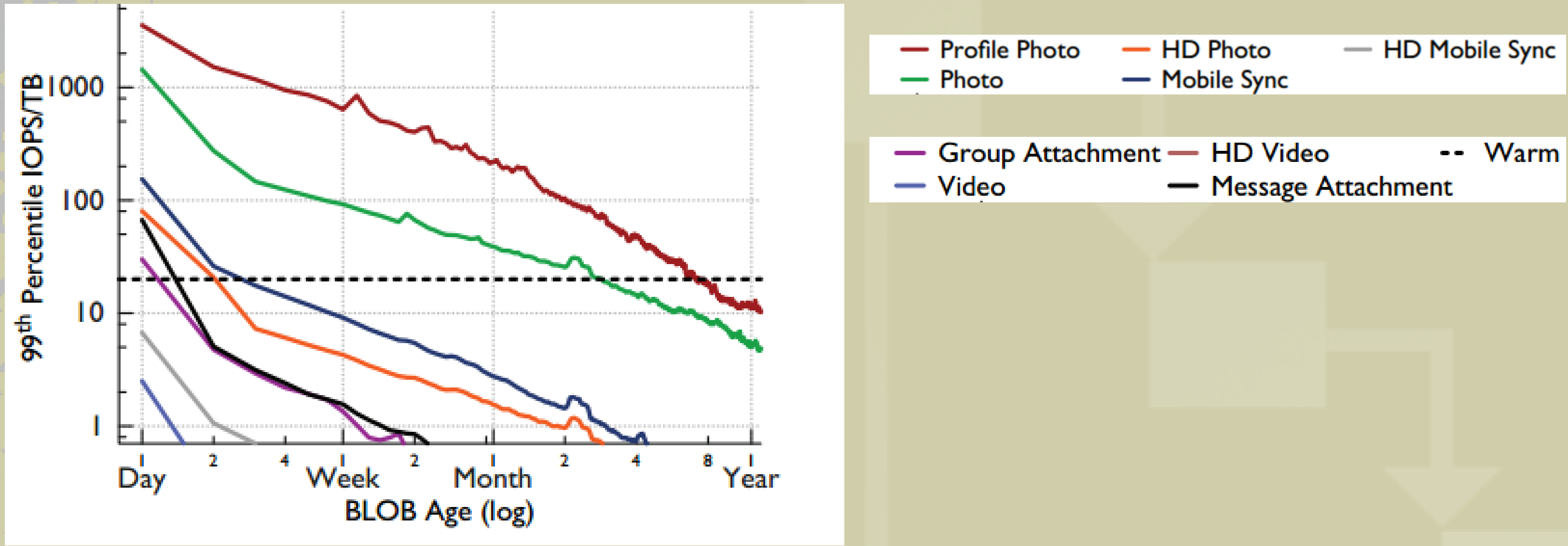
Why Warm Storage?

- Presence of temperature zones
- Two week-trace- random 0.1% reads, 10% creates, 10% deletes



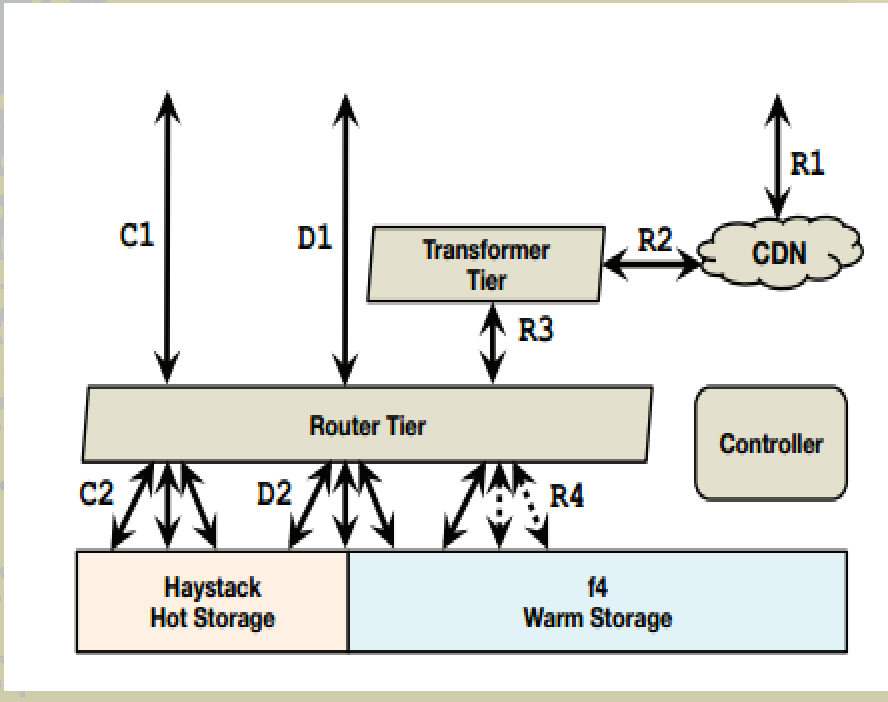
- Points in graph represent decrease in request rates by an order of magnitude

Hot or Warm?



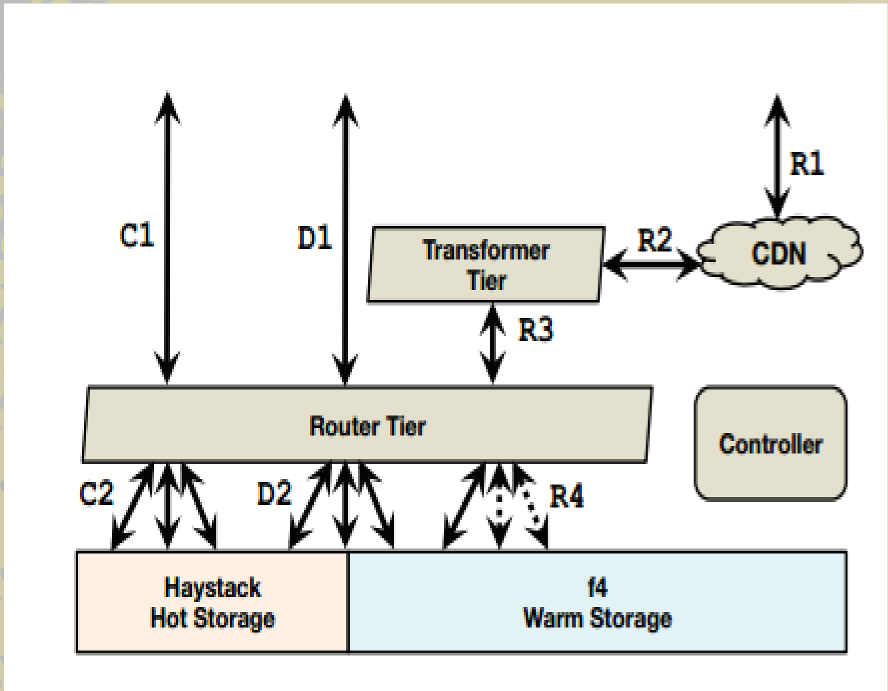
- When to move content to warm storage
- 7 of 9 content types- less than a week
- Photos- three months, Profile photos- 1 year
- Warm content continuing to grow

Overall Blob Storage Design



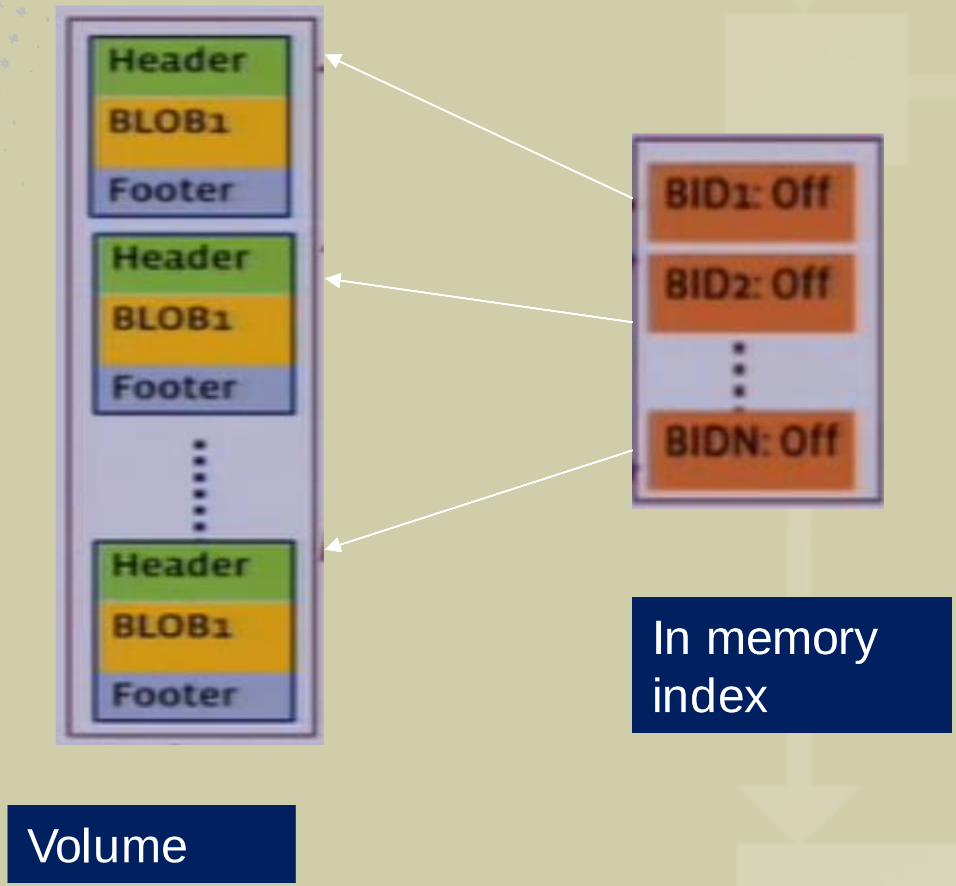
- Creates and deletes handled by Haystack
- Reads handled by either Haystack or f4

Overall Blob Storage Design



- Controller- maintenance tasks, compaction,gc
- Router Tier- hides storage implementation from clients, mapping information
- Router tier enables addition of f4
- Transformer tier- handles transformations like resizing and cropping photos, takes these away from the storage systems

Hot Storage using Haystack



- Volumes are series of blobs
- Unlocked volumes: support reads, creates and deletes
- Once 100GB reached, transitioned into locked and no longer allows creates
- Volumes contain data file, index file and a journal file

Hot Storage using Haystack

- Creates only a small number of files (~100)
- Bypasses underlying file system for most metadata access
- Minimal set of metadata for identifying BLOBs, kept in memory
- Reduces IOPS for metadata fetches
- Fault tolerance to disk, host, rack and datacenter failure through triple replication of data and hardware RAID-6(1.2X replication)
- Good fault tolerance and high throughput
- But, effective-replication-factor 3*1.2=3.6 (not suitable for warm storage, which should be storage efficient !)

Warm Storage using f4

- **Design goal 1: Storage efficiency**
- Reduce the effective-replication-factor, while maintaining high degree of reliability and performance
- Effective-replication-factor of 3.6 is too high

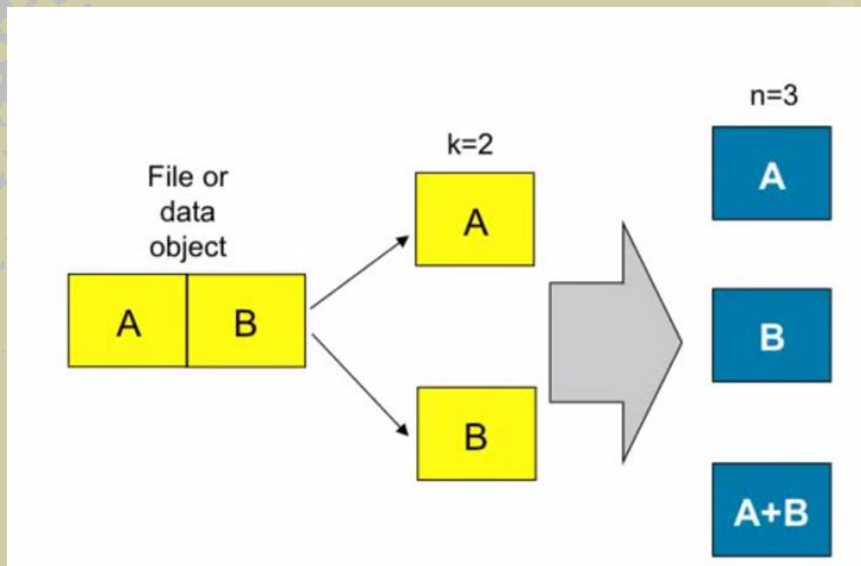
- **Design goal 2: Fault tolerance**
- Drive failure, at low single digit annual rate
- Host failure, periodically
- Rack failure, multiple times per year
- Datacenter failures, very rare

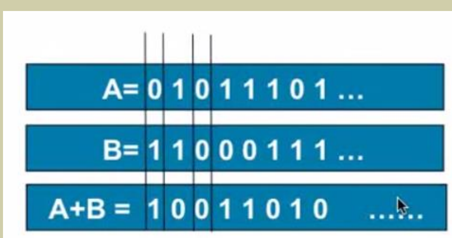
F4 Cell

- Basic building block of f4 storage
- Entirely fits into a datacenter
- Contains 14 racks of 15 hosts with 30 4TB drives per host
- Treated as a single unit of acquisition and deployment.
- Fault tolerant to disk, host and rack failures
- Because of using Reed Solomon Encoding (next slide)

- Components
  - Name node
  - Storage node
  - Backoff node
  - Rebuilder node
  - Coordinator node

Simple Erasure Coding

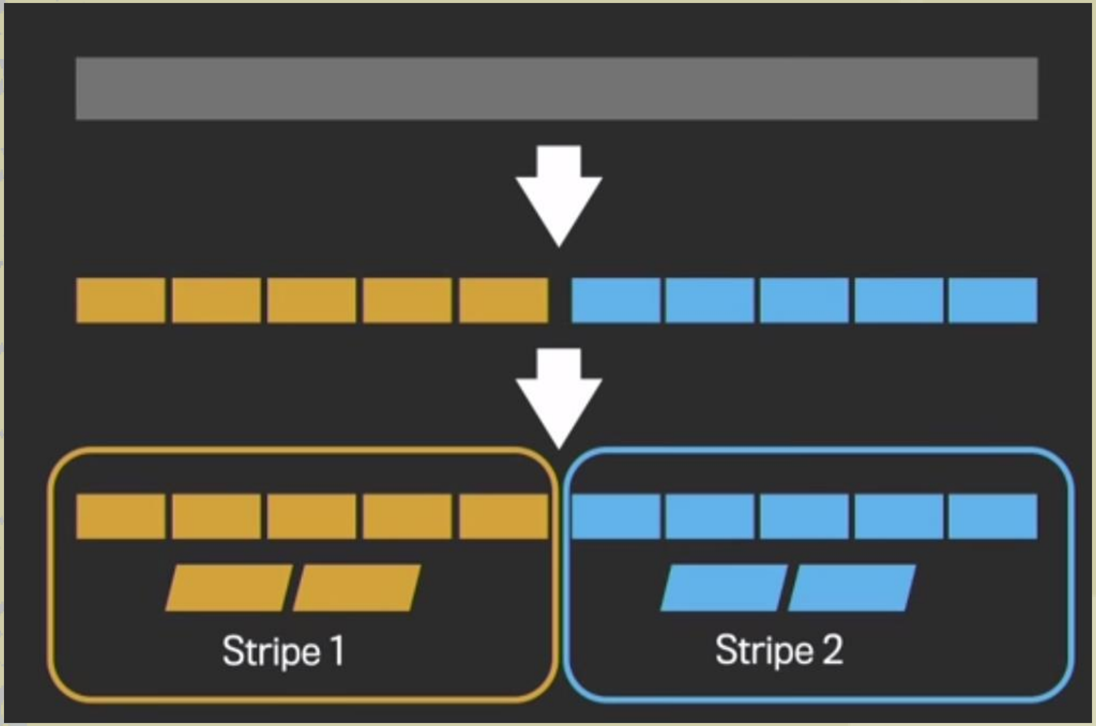




- A (n,k) erasure code provides a way to:
- Take k blocks and generate n blocks of the same size
- Any k of n sufficient to reconstruct the original k
- Reed-Solomon a type of erasure code

Reed Solomon Encoding

- A popular eraser coding technique
- Has effective replication factor of 1.4x
- In production, RS(10,4) is used
- Tolerates 4 rack failures, and therefore 4 disks/hosts



10 GB Volume

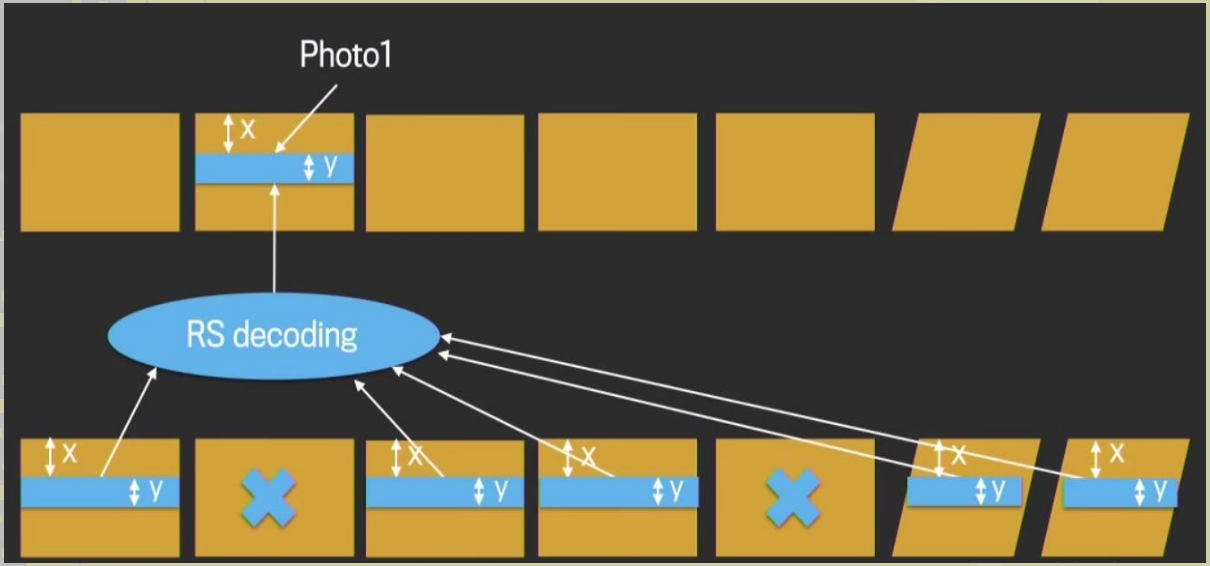
1GB Data blocks

Data blocks

Parity blocks

Reed Solomon Rebuilding

- Recovery of a block in case of failure
- Entire blocks not needed for recovery of a BLOB

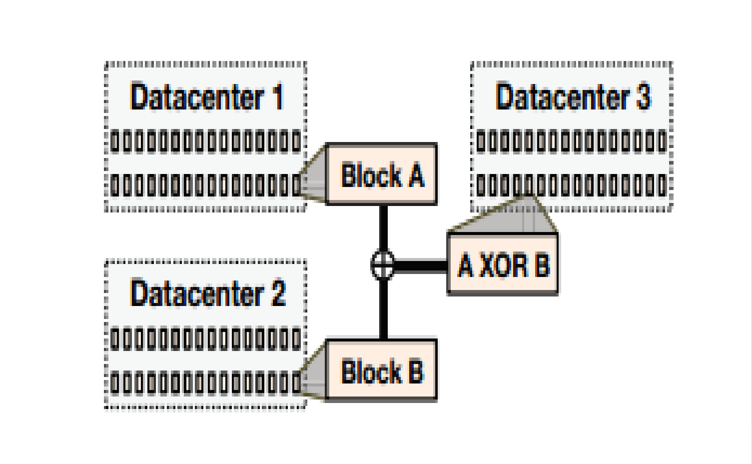


X- Offset of photo1 in its block

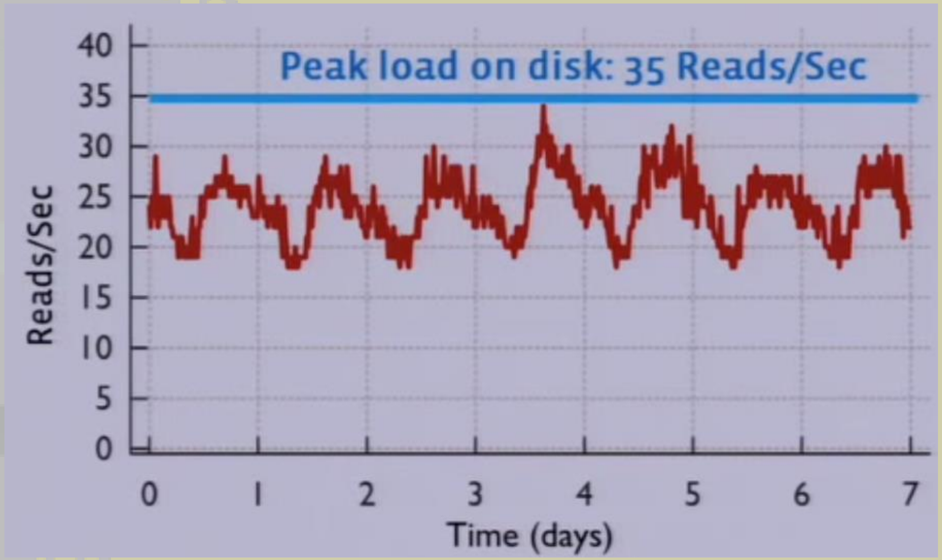
Y- Size of Photo1

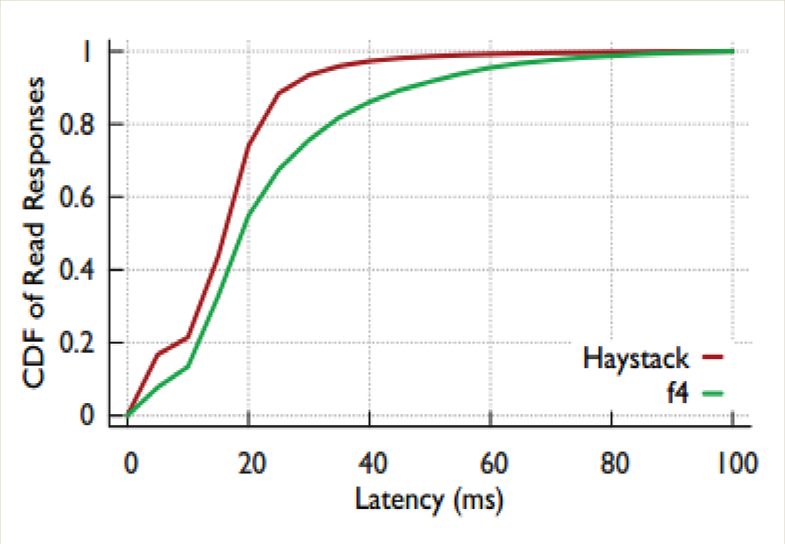
Geo-replication

- Initially data center level double replication, resulting in effective replication factor of 2.8
- Using XOR coding and accepting the tradeoff of reduced throughput for BLOBs stored at failed data center, effective replication factor reduced to 2.1



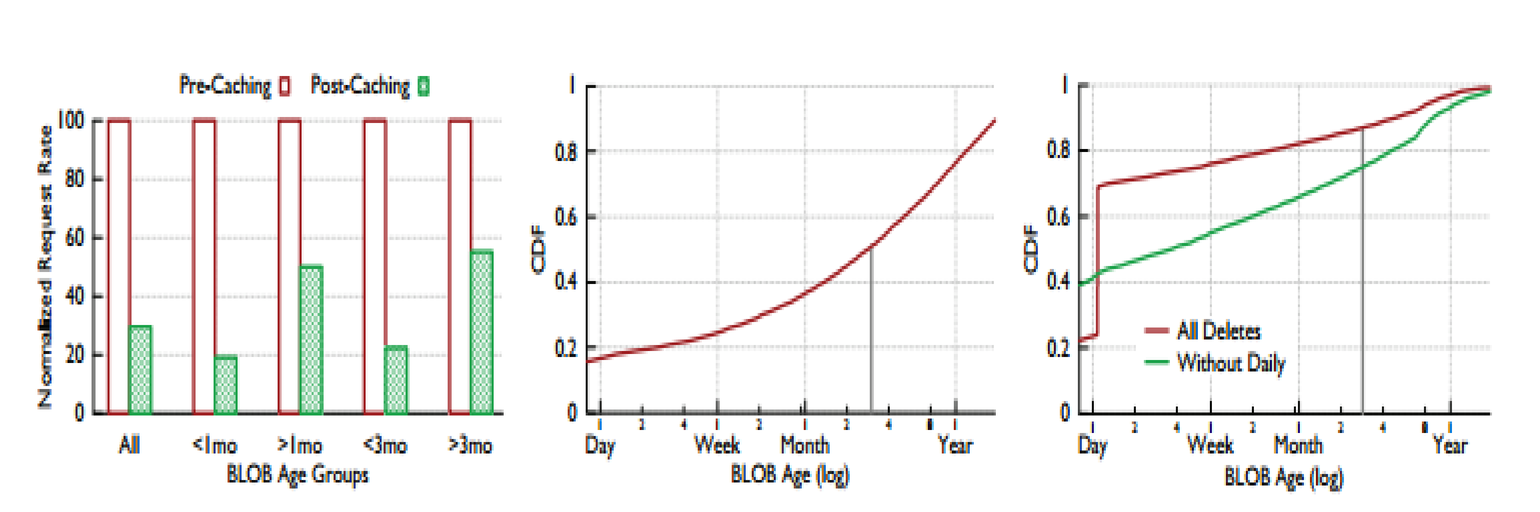
Evaluation





- Peak load on the most overloaded cluster
- Still < 80 Reads/sec

Evaluation



Pre-Caching
Post-Caching
Normalized Request Rate
100
80
60
40
20
0
All
<1mo
>1mo
<3mo
>3mo
BLOB Age Groups
CDF
1
0.8
0.6
0.4
0.2
0
Day
Week
Month
Year
BLOB Age (log)
All Deletes
Without Daily

Evaluation summary

- f4 able to handle near worst case loads with lower throughput
- f4 is resilient to failure at all levels
- f4 saves storage:
  - Current corpus 65PB
  - Saves 39PB for 2.8 erf
  - Saves 87PB for 2.1 erf
- f4 provides low latency for reads:
  - Less than 30ms for 80%
  - Less than 80ms for 99%

Comments or criticism

- Need to factor in the popularity of the profile, not solely based on the age
- What about the overhead of reconstruction done by backoff and rebuilder nodes
- Why is the transition to warm storage permanent?
- Ambiguous about the exact time period in which various BLOB types are moved to f4 from Haystack