Haystack full of needles.

Beaver, D., Kumar, S., Li, H.C., Sobel, J., and Vajgel, P.: “Finding a Needle in Haystack: Facebook's Photo Storage,” in Proceedings USENIX OSDI 2010.

Introduction

- 65 billion photos (in 4 copies)
- 260 billion images
- 20 petabytes of data
- 1 billion photos each week (~60 terabytes)
- 1 million views per second (at peak)

System specification

System stores specific data which is:

- Written once
- Read often
- Never modified
- Rarely deleted

Requirements

- High throughput
- Low latency
- Fault-tolerant
- Cost-effective
- Simple

Problem?

All exisiting storage system performed poorly on facebook workload.

Background

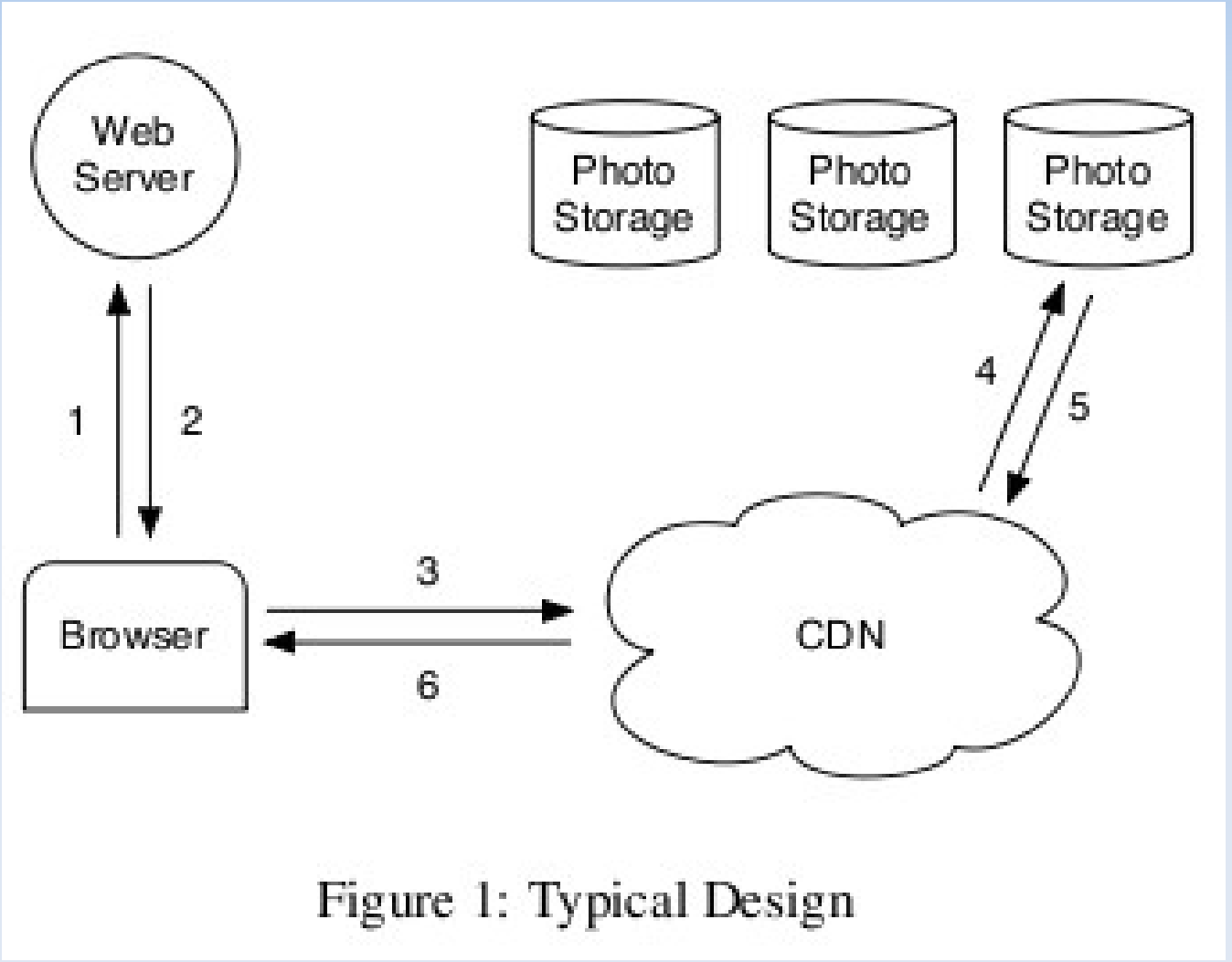


Figure 1: Typical Design

NFS-based Design

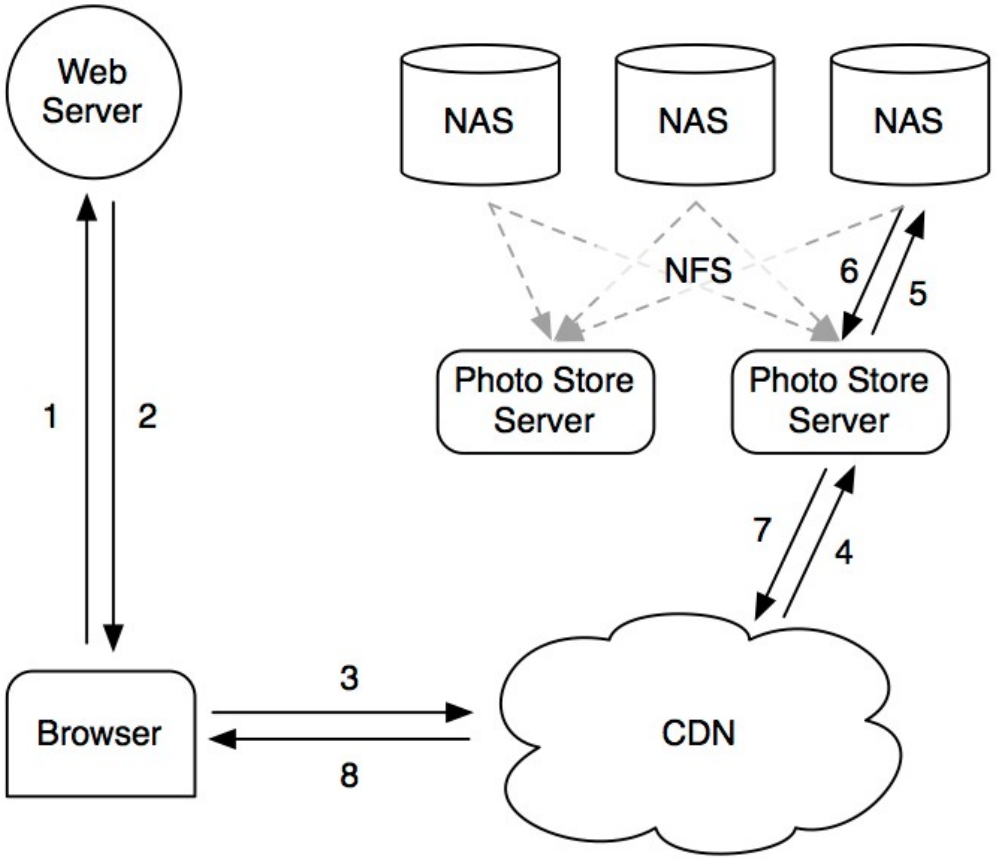


Figure 2: NFS-based Design

NFS-based Design

- NFS directories – large directory blockmap.

  10 op. before reducing directory size.

  After reducing – 3 op. :
  - Read directory metadata
  - Load inode
  - Read file content
- CDNs effectively serve cached photos (new ones) Problem? Long tail!

Solution

- Store files metadata in main memory
- Problem? Inode size...
  - xfs_inode_t takes 536 bytes
- So let's make metadata smaller!

Haystack overview

- Haystack Store
- Haystack Cache
- Haystack Directory

Haystack overview

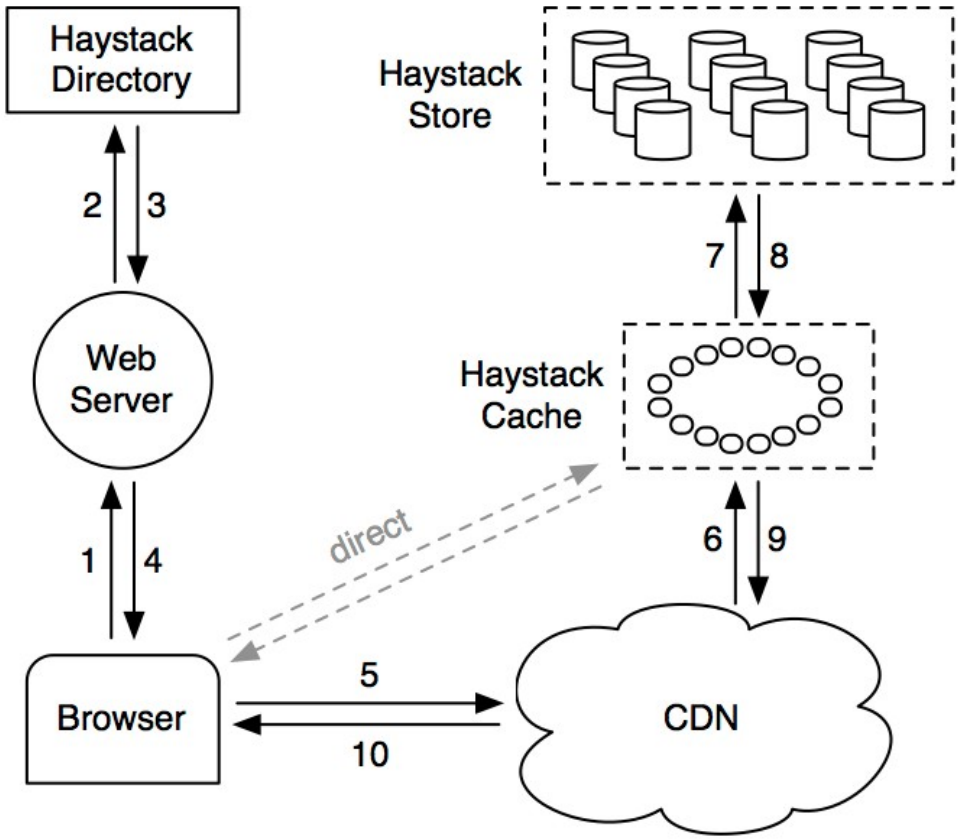


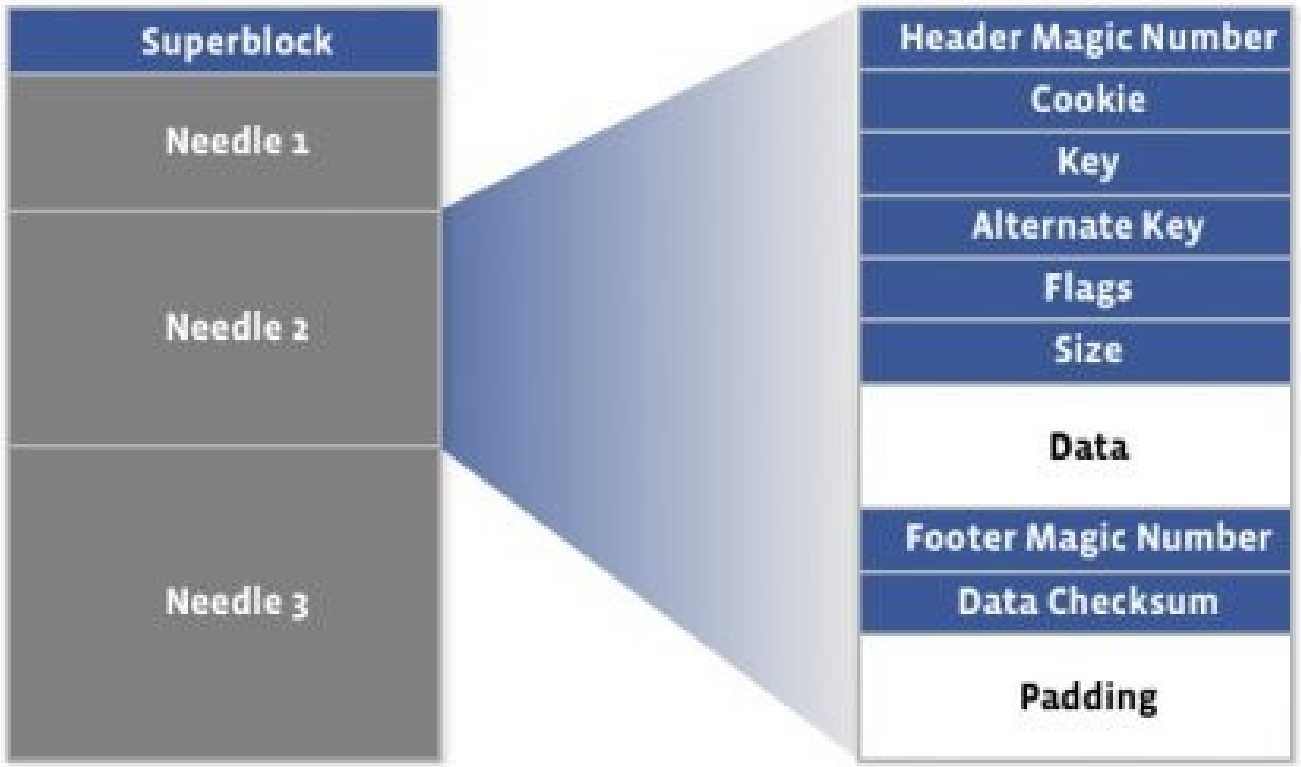
Figure 3: Serving a photo

Haystack Store

- Manages filesystem metadata for photos
- Storing data in few copies

- Why is effective?

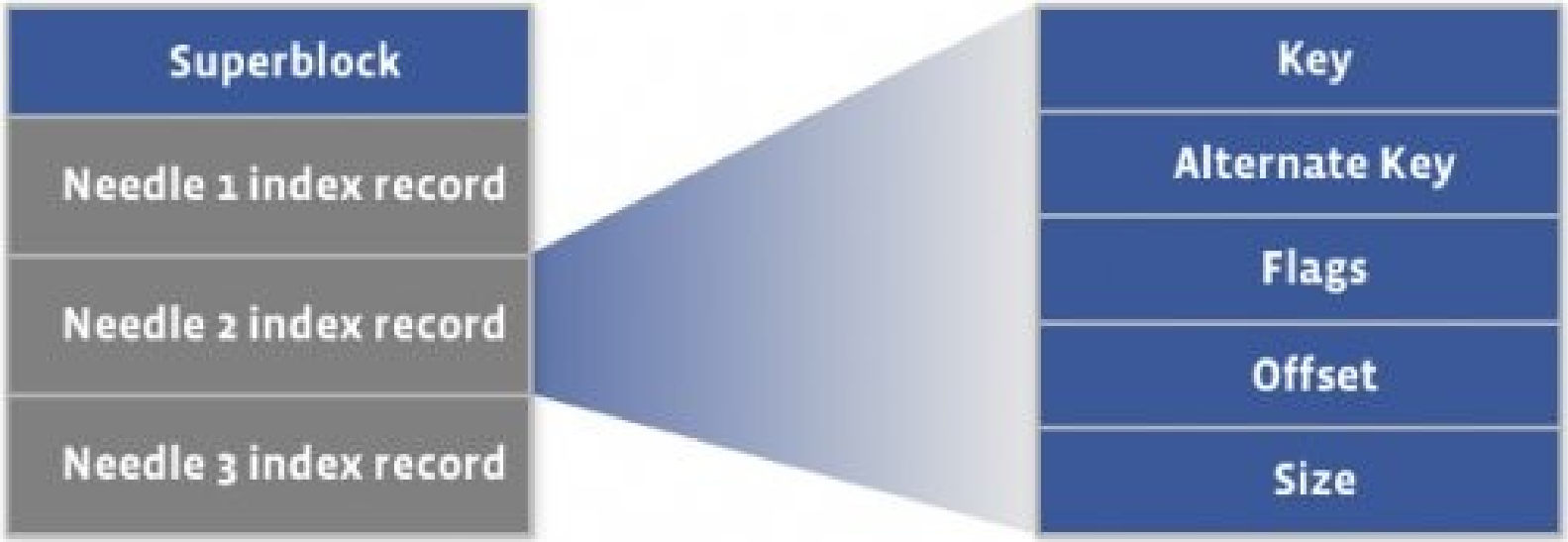
Store layout



Needle metadata

Header Magic Number	Magic number used to find the next possible needle during recovery
Cookie	Security cookie supplied by the client application to prevent brute force attack
Key	64-bit object key
Alternate Key	32-bit object alternate key
Flags	Currently only one signifying that the object has been removed
Size	Data size
Footer Magic Number	Magic number used to find the possible needle end during recovery
Data Checksum	Checksum for the data portion of the needle
Padding	Total needle size is aligned to 8 bytes

Store index


Superblock
Needle 1 index record
Needle 2 index record
Needle 3 index record
Key
Alternate Key
Flags
Offset
Size

Store index

- Can be stored in main memory
- Despite asynchronous update can be updated on reboot (or on photo demand)

Store usecases

- Photo read
- Photo write
- Photo delete
- Photo change?

Store optimizations

- Compaction – coping (skiping deleted and altered files)
- Reducing metadata (now it is 40bytes per image)
- Batch upload. Better performance when writes/read only

Haystack Directory

- Mapping from logical volumes to physical

  (used during uploads and for URL construct)
- Balances writes across logical volumes and reads across physical
- Determines whether use CDN or not
- Identifies read-only volumes.

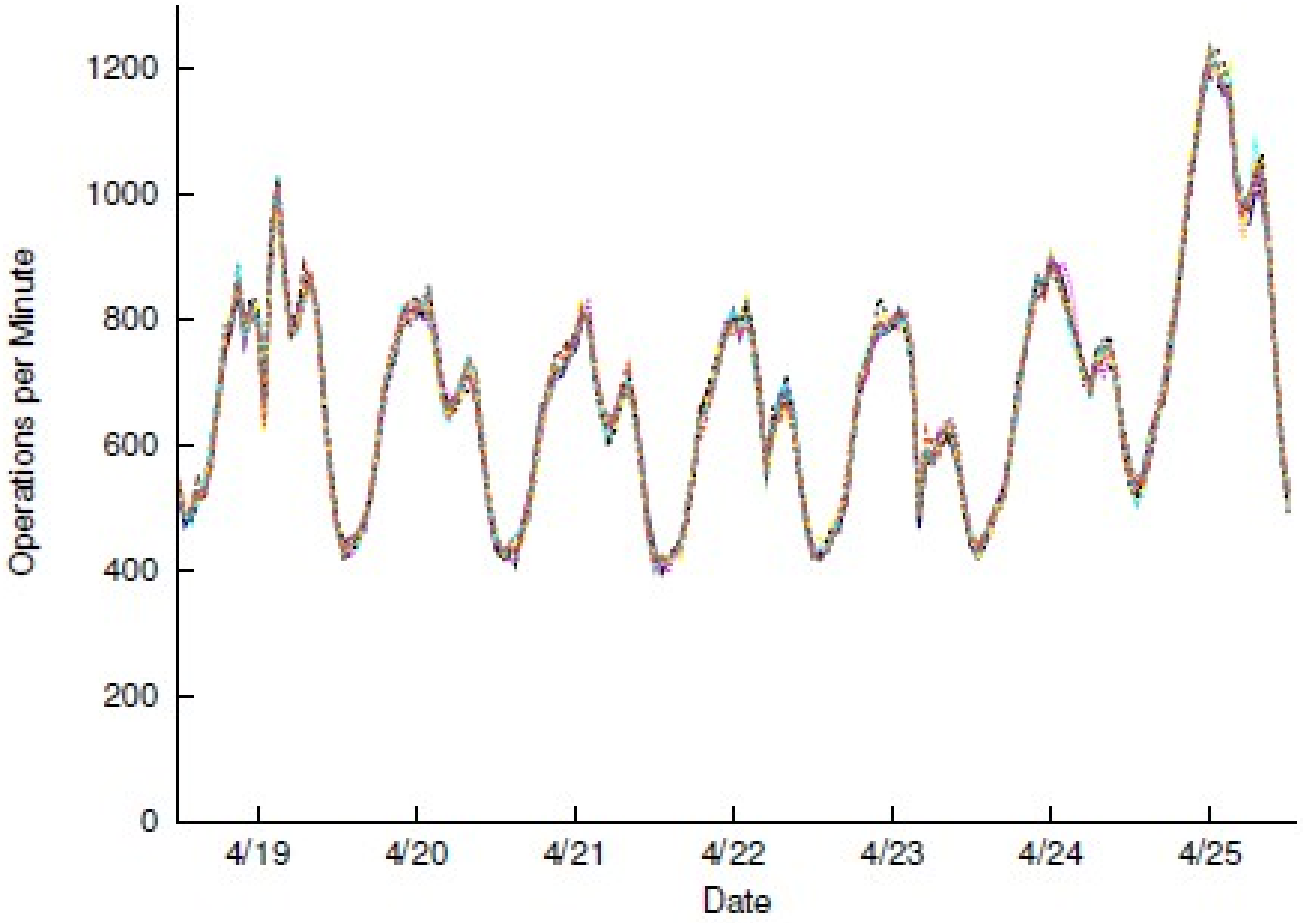
Haystack Cache

- Extended cache
- Cache only data when:
  - Request is direct form user
  - Photo is fetched form write enabled vloume
- Used to protect write-enabled volumes

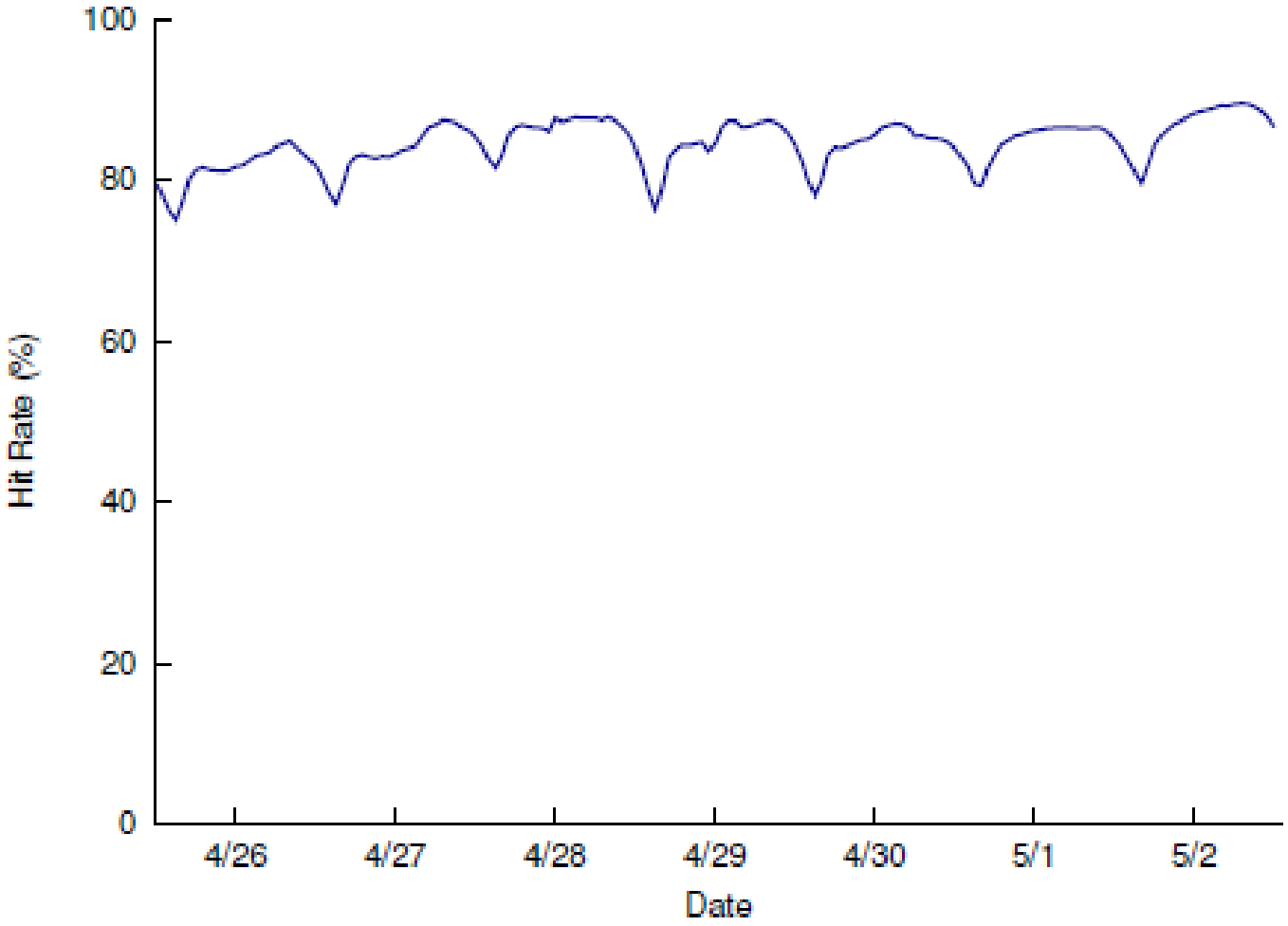
Evaluation: Daily traffic

Operations	Daily Counts
Photos Uploaded	~120 Million
Haystack Photos Written	~1.44 Billion
Photos Viewed	80-100 Billion
[*Thumbnails*]	10.2 %
[*Small*]	84.4 %
[*Medium*]	0.2 %
[*Large*]	5.2 %
Haystack Photos Read	10 Billion

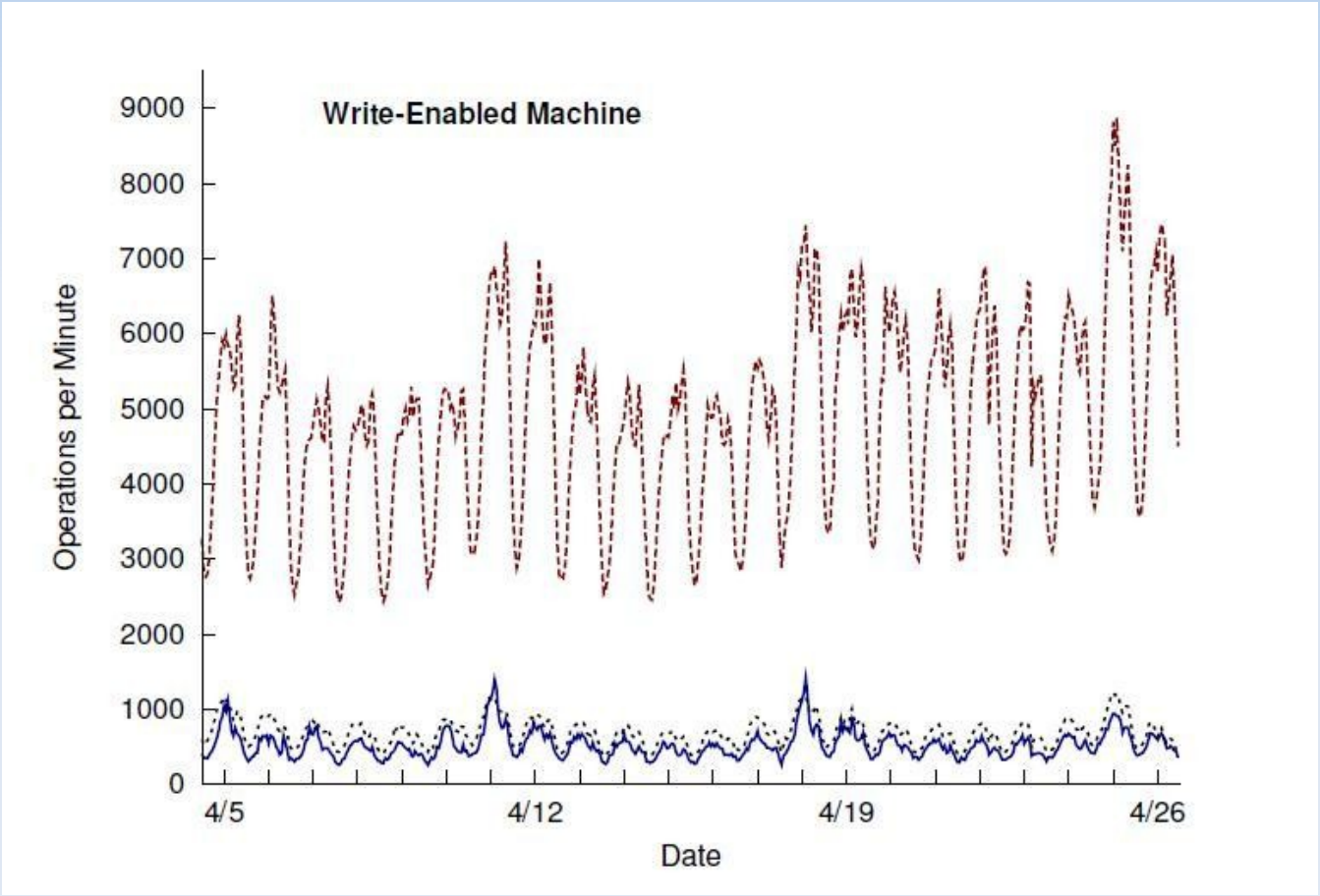
Evaluation: Directory



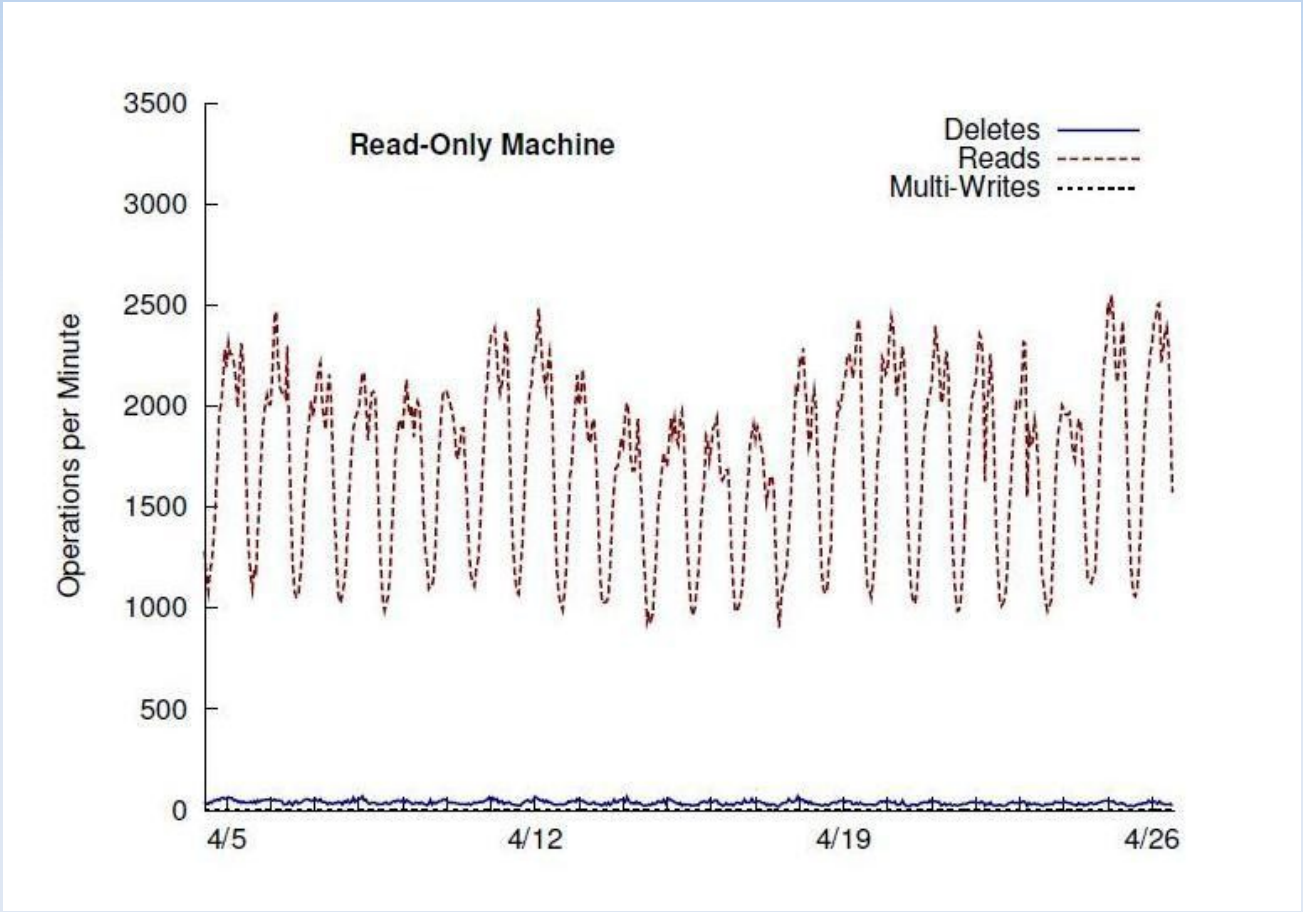
Evaluation: Cache



Evaluation: write-enabled machine



Evaluation: read-only machine



Concusion

- System handling long tail content (crucial in social network)
- Fault-tolerant
- Higher troughput
- Less cost
- Simple
- Scalable

Questions?

Thank you!

Przemysław Spodymek