Instagram

Sowmya
study notes

→ Functional requirements:

* view/ upload/ download photos
* search for images using titles
* users can follow other users
* generate and display newsfeed for each user

→ Non-functional requirements:

* Availability (consistency can take a hit)
* Reliability (photos uploaded cannot be lost)
* Acceptable latency for generating news feed (eg. 200ms)
* Low latency while viewing photos

→ Note:

* Read-heavy system
* Huge photo-base, so efficient storage required

→ Capacity estimation:

500M users in total
1 M active / day

Traffic:

1M * 2 (avg) photos/user ≃ 2M/day ⇒ $\frac{2M}{24*60*60}$ ≃ 23 ph/sec new

1M * 20 (feed-size) ≃ 20M/day ≃ 230 ph/sec (reads)

Storage:

2M * 200 Kb ⇒ 400 GB for photos/day
400GB * 365 * 10 ≃ 1425 TB/10 years

Sowmya Study notes

Bandwidth: ph/sec ph.size

(incoming) ⇒ 23 * 200 Kb ≃ 4.6 Mbps

(outgoing) ⇒ 230 * 200 Kb ≃ 46 Mbps

not considering CDN

Cache: Newsfeed cache + profile cache + suggestion cache etc. + globally distributed cache servers (CDN)

Newsfeed cache ≃ 0.2 * 20M * 200Kb

≃ 4M * 2×10^5

≃ $4 \times 10^6 \times 2 \times 10^5$

≃ 8×10^{11} ≃ 800 GB/day

avg estimate for each newsfeed cache

→ **DB Schema:** User, photo, userFollow

100 bytes

User
UserID → pk
Email
Name
createdAt
LastLogin
Active
DOB

Photo
PhotoID → pk
user ID
CreatedTime
Photopath
photo Latitude
photoLongitude
userlatitude
userLongitude

500 bytes

UserFollow
followerID] pk
followeeID]

8 bytes

index on photoID & creationDate to fetch latest photos first

Soumya
study notes

RDBMS ? since we require joins?
Can they scale ?

Store photos in distributed file storage like HDFS or S3

User → 100 bytes * 500M ≃ $5 \times 10^8 \times 10^2$
≃ 50 GB

Photo → 500 bytes * 2M photos/day ≃ 1000×10^6
≃ 1 GB/day * 365 * 10
≃ 3.65 TB / 10 years

UserFollow → 500M * 500 followers * 8 bytes
≃ $2 \times 10^6 \times 10^6$
≃ 2 TB

≃ 2TB + 3.65TB + 50GB ≃ 5.7TB

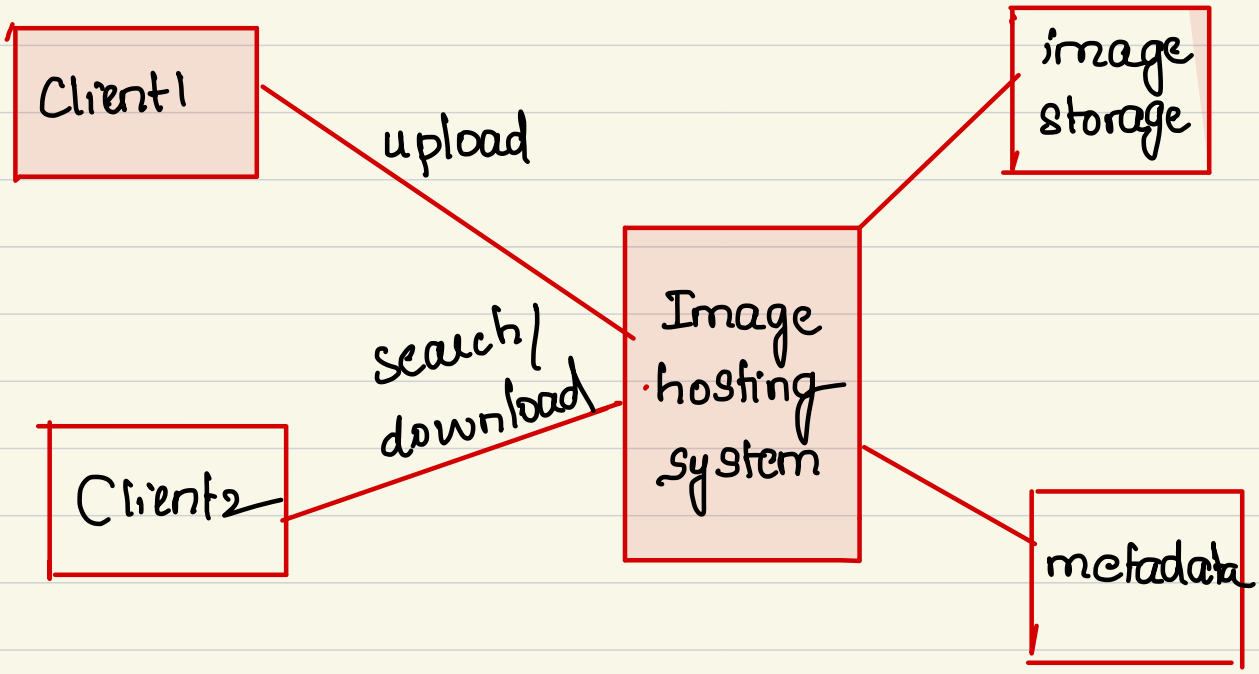
Key-value pairs
PhotoID: [photoLocation, userLocation...]
userID: [Name, email, createdAt...]
userID: [list of photoIDs]
userID: [list of followee IDs]

AP data-stores like Cassandra

System APIs:

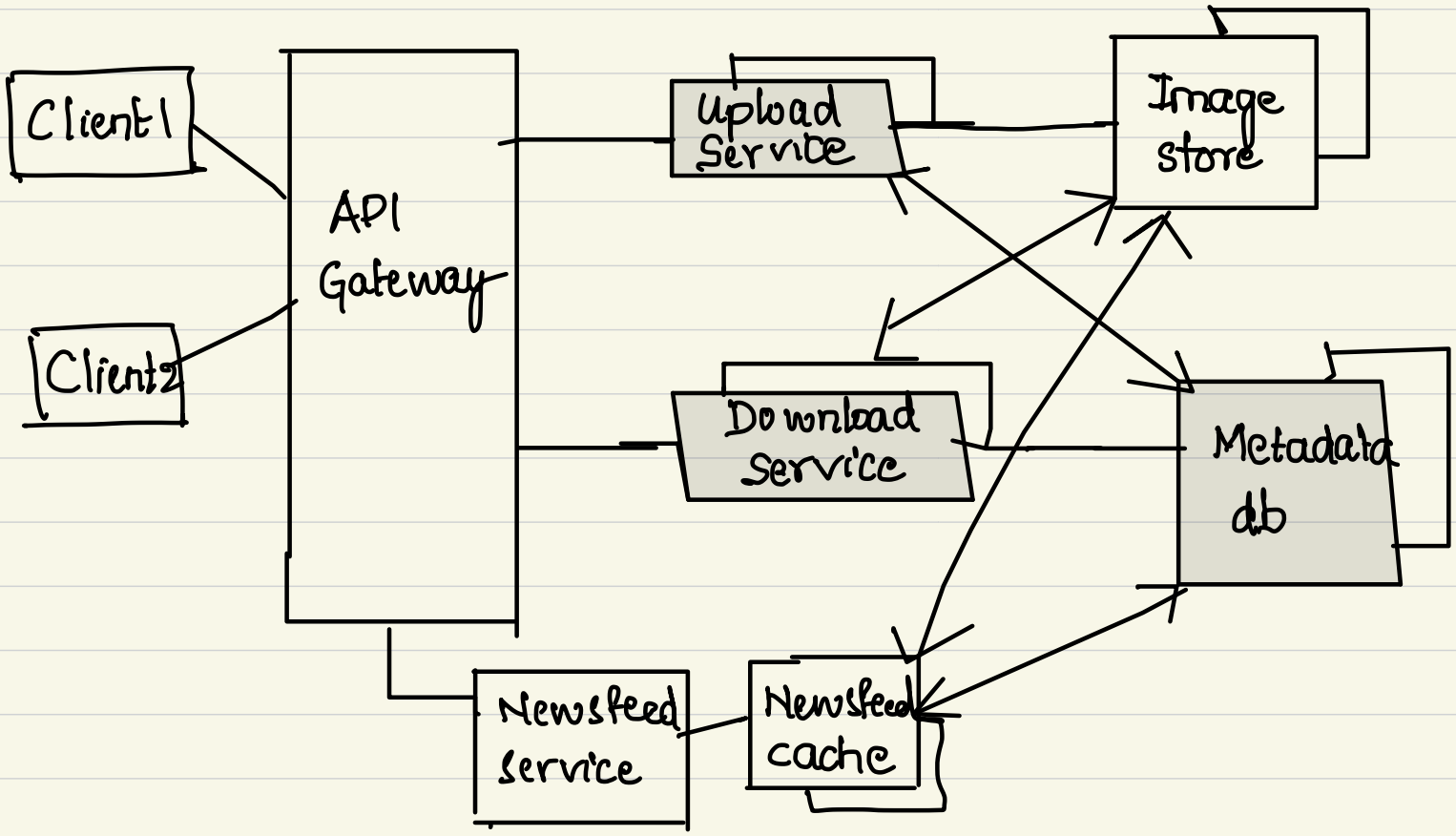
uploadImage(userID) generateNewsfeed(userID)
viewImage(userID, photoID)

Sowmya
study notes



HIGH LEVEL DESIGN

Component Design:



LOW-LEVEL DESIGN

Sowmya
Study notes

Trade-offs

Partitioning

a) Partitioning based on userID : photos of each user on same shard
Generating photoIDs? every shard → auto increment sequence → append shardID ⇒ generates unique photoID
↓
Hot users?
Unbalanced shards?
high latency of shard due to heavy load

b) Partitioning based on photoID: generate unique photoID & then shard

Generating photo IDs? KGS similar to tinyUrl

Newsfeed generation

* Pull: Clients can pull newsfeed contents from server manually → problem? (manual poll)
* Push: server can push content to user whenever available → problem? user follows lot of people or celebrities (frequent pushes)
* Hybrid: Pull for people who follow more people/celebs
push for everyone else (or)
push updates to people in batches