3) SYSTEM DESIGN OF INSTAGRAM

1) Requirements & Clarifications:
(i) Use can upload photos / view photo
(ii)User can follow Another people
(iii) News Feed which contains top photos of people followed by user
(iv) Availability
(v) Total 500M users with Daily 1M active users
(vi) Daily 2M photos uploaded.
(vii) Read Write ratio 8:1

2) Capacity Estimation :

(i) Traffic/ Query Per Second (QPS) ??
Write(QPS) :
2M / 24(hours) * 3600(seconds) = 12 / sec

Read (QPS) :
8 * 2M / 24 * 3600 = 185 / sec

(ii) Storage Required ??
Assuming 10 years and each object is of 200KB
2M * 10 (years) * 12 (months) * 30 (days) * 200 KB = 1245 TB

(iii) Network Bandwidth ??
Write :
12 (uploads per second) * 200 (KB)
= 2.4 MB / sec

Read :
185 (reads per second) * 200 (KB)
= 27 MB / sec

(iv) Memory Estimation (caching) ??
Considering 80:20 rule , that is 20% Redirecting generating 80% traffic

0.2 * 16M * 200 (KB)
= approx 640 GB

note: less than 640GB will be needed, as majorly requests wiill be duplicates

3) System Interface Definition

(i)upload(userId, photo,text,photo_latitude, photo_longitude,current_date_time)

(ii)viewUserPhotos(userId, current_date_time)

(iii)viewUserFeed(userId, current_date_time)

4) Define Data Modeling
Observations are as below
(i) write heavy and also Read heavy
(ii) billions of records to be stored
(iii) relationship for user - photos & user - follows
(iv) object size is 200KB

we can have NO-SQL database considering above and due to horizontal scaling.

USER
userId int (pk)
username varchar 50
creationDate dateTime
lastLogin dateTime

PHOTO
photoId int (pk)
photoPath varchar 500
userid int 10 (fk)
photoText varchar 500
photoLongitude int
photoLatitude int

FOLLOWS
userId int
followingUserId int

We can use distributed Key-DB for PHOTOS which has key as photoId and value as other parameters of photos.

For USERPHOTO we can use wide - column database like Cassandra which has key as userId and other columns as photoIds.

For USERFOLLOWS same scheme as above.

Data Sharding:
We can shard photoIds and we can generate photoIds by (KGS) or dedicated database which gives incremented IDs

For UserFeeds we can do sharding based on photoId and Epoch time

5) High Level And Detailed Design

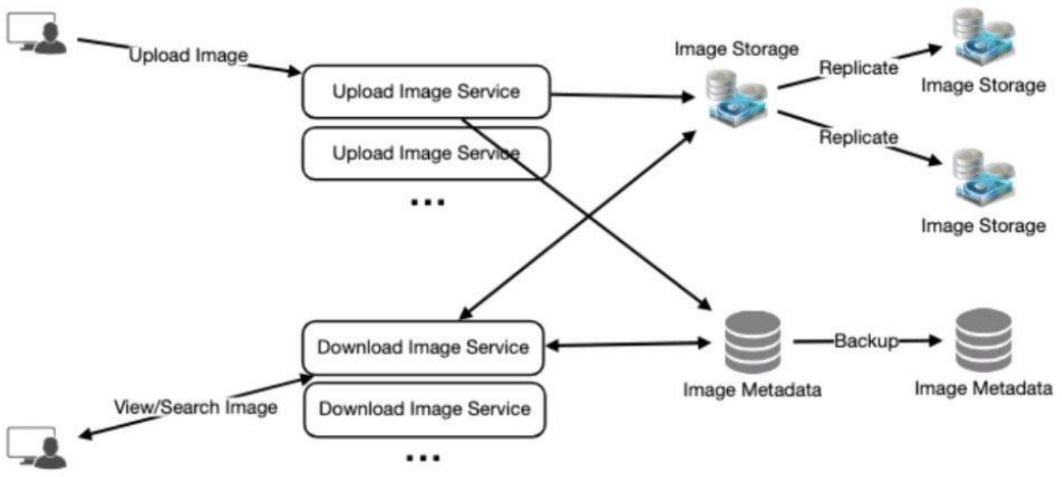
Uploading Photos Logic:
Client request --> Application server --> Object storage (e.g amazon s3) and then insert into meta-data store (e.g MySql or Cassandra)

UserFeed Logic:
Client request --> Application server --> Object Store & Meta Data store --> ranks them based on factors (recent,likes...etc) and returns photos

For Fetching News Feed to Users, below approach can be used.
1) Push
2) Pull
3) Hybrid

For READ requests, geographical distributed photo cache servers and CDN needs be used.

(LRU) cache eviction policy can be applied.


Upload Image
Upload Image Service
Upload Image Service
...
Image Storage
Replicate
Image Storage
Replicate
Image Storage
Download Image Service
Download Image Service
...
View/Search Image
Image Metadata
Backup
Image Metadata