Captsone Project

INSTAGRAM NEWSFEED DESIGN

Functional Requirements

1. Create Posts (Text, Image, Video)
2. Like/Comment
3. News Feed
4. Notifications on Post Engagement

Non Functional Requirements

1. Availability: The system should be highly available
2. Eventual Consistency: If a user posts something, it's okay if it doesn't show up immediately but appears within a few seconds (1-2 seconds)
3. Low Latency: The news feed should load within 1 or 2 seconds
4. Scalability:
    a. The system should support 500 million Daily Active Users
    b. The system should support 3 billion Monthly Active Users

Capacity Estimation

Monthly Active Users: 3 Billion

Daily Active Users: 500 Million

Throughput Estimation

- **Writes:**
    - Creating Posts
    - Liking
    - Commenting

Code block

```
Assuming 10% of DAU create posts in a day
Total DAU = 500 Million
```

```
10% OF DAU = 50 Million users create posts daily
50 Million write requests / daily = 600 writes/sec
```

- **Reads:**
  - User viewing news feed

Code block

```
Total DAU = 500 Million
Each user reads 100 posts per day
Total read requests = 500 Million * 100
50 Billion read requests / day = 600k reads/sec
```

Storage Estimation

- **Post Data**

Code block

```
Text Posts: 100KB = 0.2 * 50M *100KB = 1TB/day
Image Posts: 0.5MB = 0.6 * 50M * 0.5MB = 15TB/da
Video Posts: 20MB = 0.2 * 50M * 20MB = 200TB/day
Total Storage per day = 216TB/day
Total Storage in 10 years = 216TB * 365 * 10 = 750 PB
```

Network and Bandwidth Estimation

Data flow into our system per second (Ingress)

Code block

```
Incoming data per second = 216TB / 86400 = 2.5GB/second
```

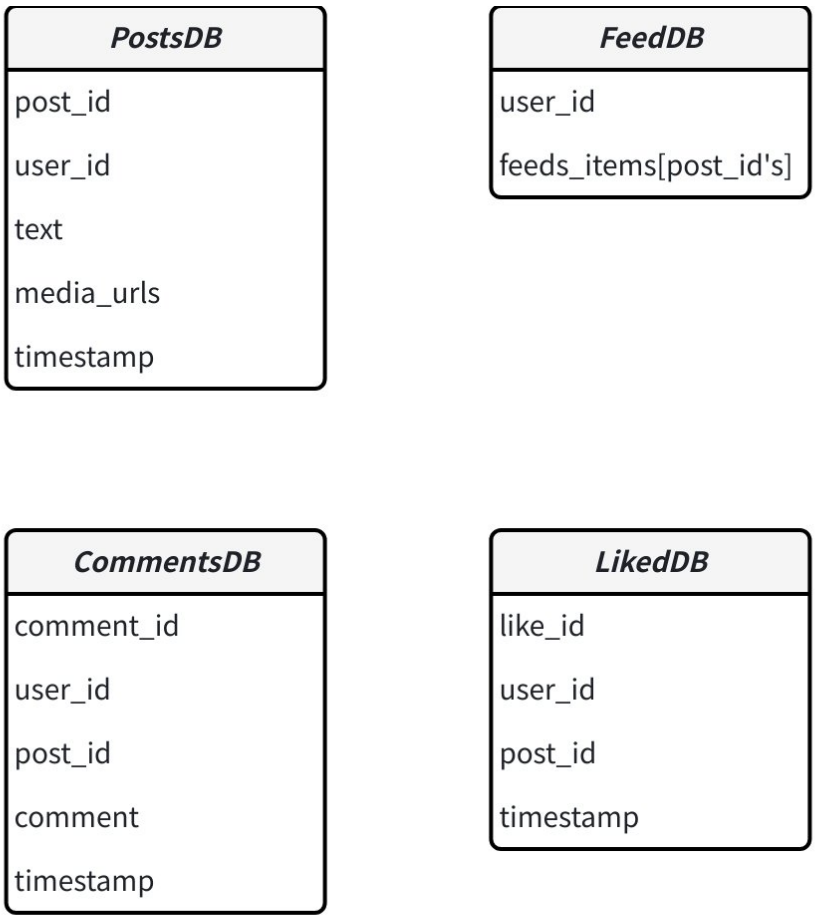
Data flow out of our system per second (Egress)

Code block

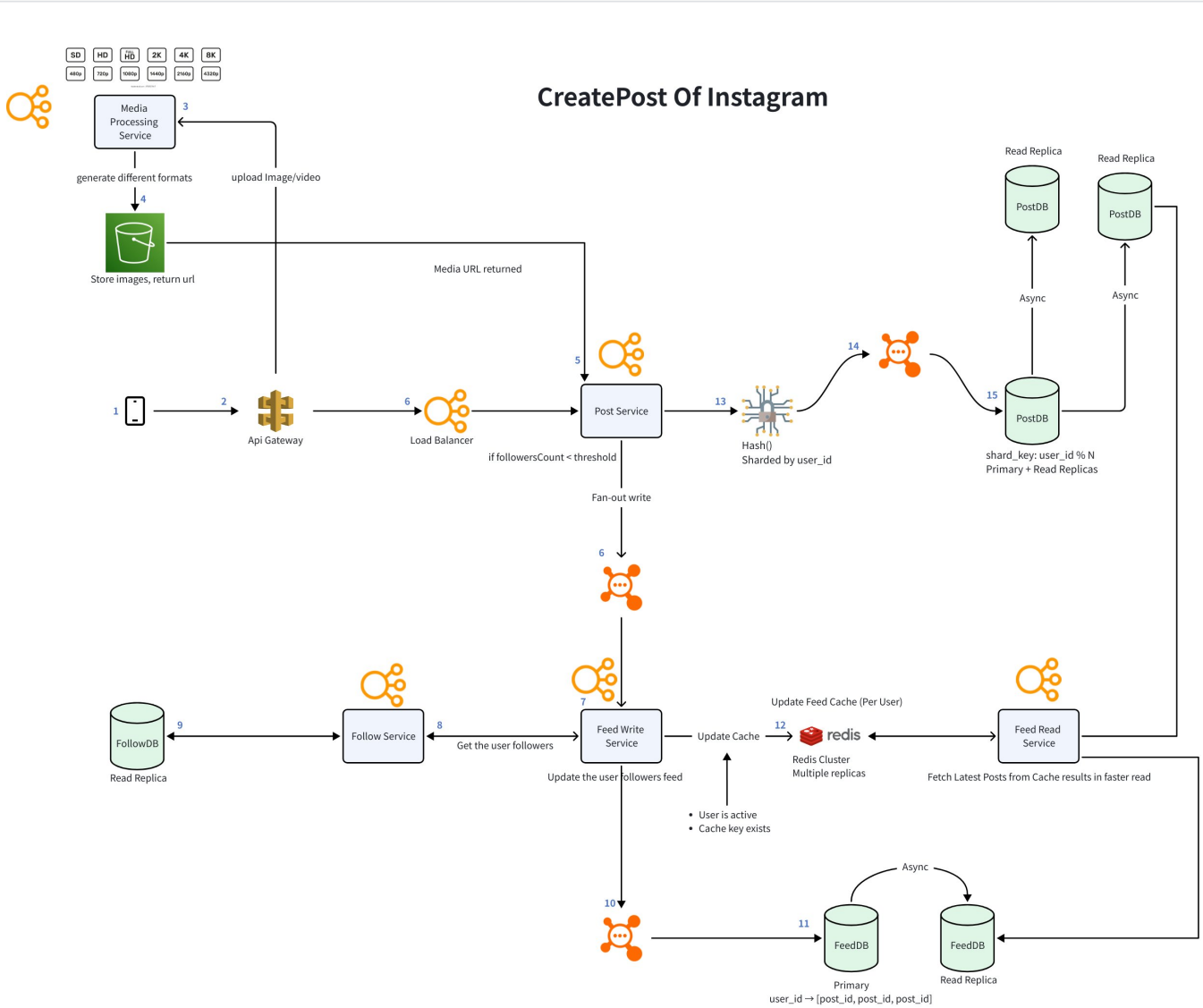
```
Total read requests per day = 50 billion
Average post size:
(0.2 × 100 KB) + (0.6 × 0.5 MB) + (0.2 × 20 MB) = 4.32 MB
```

```
Total post reads per day:
50 billion × 4.32 MB = 216 PB per day
Outgoing data per second = 216 PB ÷ 86,400 = 2.5 TB/s
```

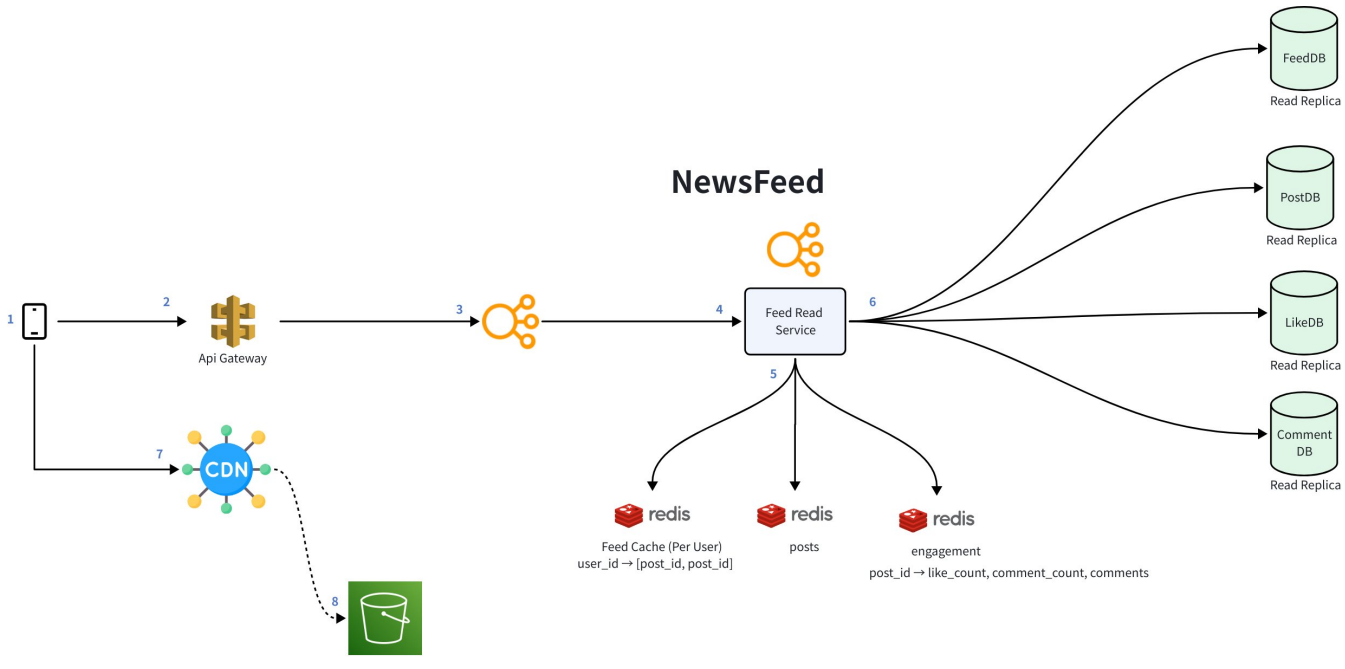
Schema



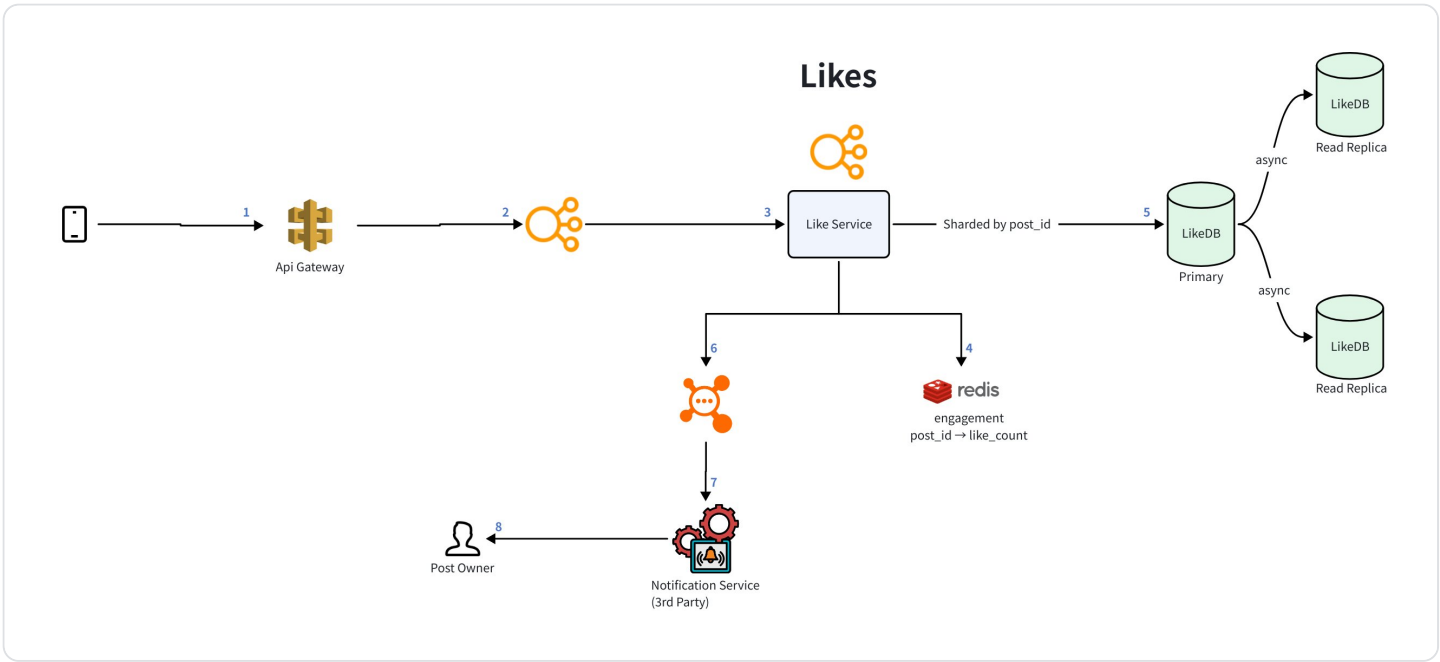
Create Post


CreatePost Of Instagram
SD
HD
HD
2K
4K
8K
480p
720p
1080p
1440p
2160p
4320p
Media Processing Service
generate different formats
upload Image/video
Store images, return url
Media URL returned
Read Replica
Read Replica
PostDB
PostDB
Async
Async
Api Gateway
Load Balancer
Post Service
Hash()
Sharded by user_id
PostDB
shard_key: user_id % N
Primary + Read Replicas
if followersCount < threshold
Fan-out write
FollowDB
Read Replica
Follow Service
Get the user followers
Feed Write Service
Update the user followers feed
Update Feed Cache (Per User)
Update Cache
redis
Redis Cluster
Multiple replicas
Feed Read Service
Fetch Latest Posts from Cache results in faster read
User is active
Cache key exists
Async
FeedDB
Primary
user_id → [post_id, post_id, post_id]
FeedDB
Read Replica

NewsFeed


NewsFeed
Api Gateway
CDN
Feed Read Service
FeedDB
Read Replica
PostDB
Read Replica
LikeDB
Read Replica
Comment DB
Read Replica
redis
Feed Cache (Per User)
user_id → [post_id, post_id]
redis
posts
redis
engagement
post_id → like_count, comment_count, comments

Likes



Comments