Requirements of Instagram's Design

Learn about the requirements and computational needs for the Instagram service.

We'll cover the following

Requirements

We'll concentrate on some important features of Instagram to make this design simple. Let's list down the requirements for our system:

Functional requirements

- **Post photos and videos**: The users can post photos and videos on Instagram.
- **Follow and unfollow users**: The users can follow and unfollow other users on Instagram.

- **Like or dislike posts**: The users can like or dislike posts of the accounts they follow.
- **Search photos and videos**: The users can search photos and videos based on captions and location.
- **Generate news feed**: The users can view the news feed consisting of the photos and videos (in chronological order) from all the users they follow. Users can also view suggested and promoted photos in their news feed.

Non-functional requirements

- **Scalability**: The system should be scalable to handle millions of users in terms of computational resources and storage.
- **Latency**: The latency to generate a news feed should be low.
- **Availability**: The system should be highly available.
- **Durability** Any uploaded content (photos and videos) should never get lost.
- **Consistency**: We can compromise a little on consistency. It is acceptable if the content (photos or videos) takes time to show in followers' feeds located in a distant region.
- **Reliability**: The system must be able to tolerate hardware and software failures.

Resource estimation

Our system is read-heavy because service users spend substantially more time browsing the feeds of others than creating and posting new content. Our focus will be to design a system that can fetch the photos and videos on time. There is no restriction on the number of photos or videos that users can upload, meaning efficient storage management should be a primary consideration in the design of this system. Instagram supports around a billion users globally who share 95 million photos and videos on Instagram per day. We'll calculate the resources and design our system based on the requirements.

Let's assume the following:

- We have 1 billion users, with 500 million as daily active users.
- Assume 60 million photos and 35 million videos are shared on Instagram per day.
- We can consider 3 MB as the maximum size of each photo and 150 MB as the maximum size of each video uploaded on Instagram.
- On average, each user sends 20 requests (of any type) per day to our service.

Storage estimation

We need to estimate the storage capacity, bandwidth, and the number of servers to support such an enormous number of users and content.

The storage per day will be:

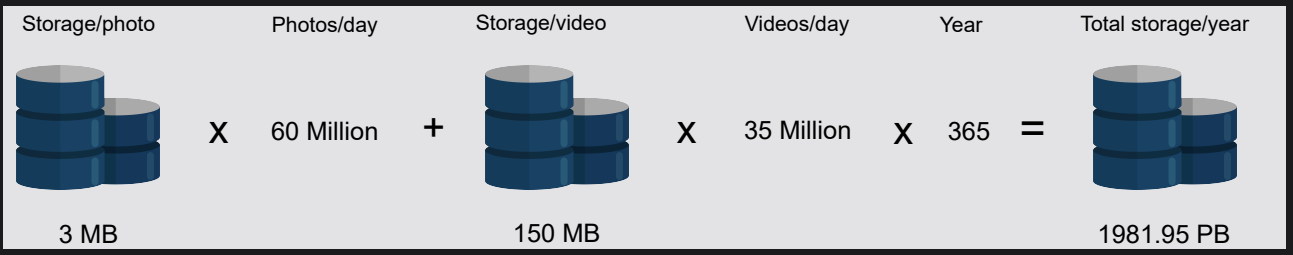
60 million photos/day * 3 MB = 180 TeraBytes / day

35 million videos/day * 150 MB = 5250 TB / day

Total content size = 180 + 5250 = 5430 TB

The total space required for a year:

5430 TB/day * 365 (days a year) = 1981950 TB = 1981.95 PetaBytes



Total storage required by the Instagram storage system for a year

Besides photos and videos, we have ignored comments, status sharing data, and so on. Moreover, we also have to store users' information and post metadata, for example, userID, photo, and so on. So, to be precise, we need more than 5430 TB/day, but to keep our design simple, let's stick to the 5430 TB/day.

Bandwidth estimation

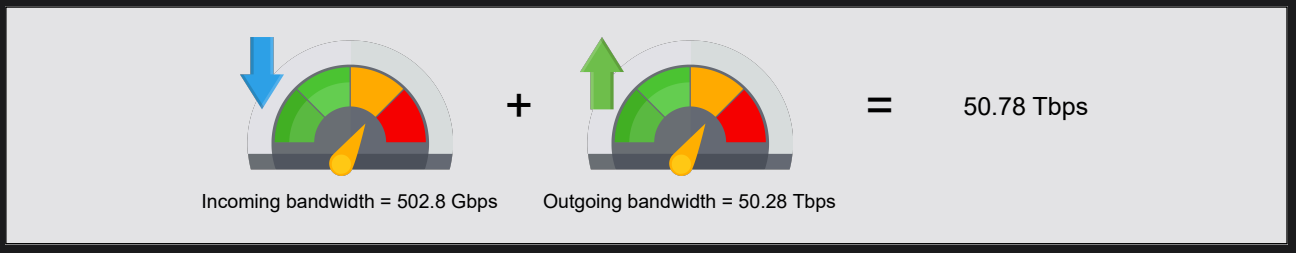
According to the estimate of our storage capacity, our service will get 5430 TB of data each day, which will give us:

5430 TB/(24 * 60* 60) = 5430 TB/86400 sec ~= 62.84 GB/s ~= 502.8 Gbps

Since each incoming photo and video needs to reach the users' followers, let's say the ratio of readers to writers is 100:1. As a result, we need 100 times more bandwidth than incoming bandwidth. Let's assume the following:

Incoming bandwidth ~= 502.8 Gbps

Required outgoing bandwidth ~= 100 * 502.8 Gbps ~= 50.28 Tbps



Total bandwidth required

Outgoing bandwidth is fairly high. We can use compression to reduce the media size substantially. Moreover, we'll place content close to the users via CDN and other caches in IXP and ISPs to serve the content at high speed and low latency.

Server count

We will estimate server count in two ways—assuming a uniform request distribution and for the peak load. In the first method, we will assume that all the requests are distributed uniformly over a day. In some sense, that might be considered the best case for our service because the load will be ideally distributed. In the second method, we assume that all the requests of a day show up simultaneously. Such a calculation gives us an upper bound on the number of servers.

One can argue that both ways might be far from the reality. However, these calculations give us lower and upper bounds on the server estimates. We can tighten these bounds by changing our assumptions. For example, if we know (or our interviewer tells us) that no more than 20% of the daily requests can show up simultaneously, then peak server estimates will change accordingly.

Server count assuming uniform request distribution

We need to handle concurrent requests from 500 million daily active users, which are uniformly distributed over 24 hours. Let's assume that a typical Instagram server handles 64,000 requests per second.

- Requests by each user per day = 20
- Total requests per day = 500 million * 20
- Requests per second = (500 million * 20) / (24 * 60 * 60) ≈ 116 K

The required servers are estimated as follows:

$$Servers\ needed\ for\ uniform\ requests = \frac{Number\ of\ requests/second}{RPS\ of\ server}$$

$$Servers\ needed\ for\ uniform\ requests = \frac{116\ K}{64,000} \approx 2\ servers$$

This calculation shows that we need only 2 servers to handle queries in our Instagram system if the request load is uniformly distributed.

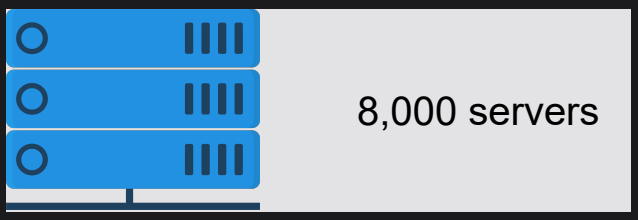
Number of servers required for the Instagram system under unifrom request distribution assumption

Server count assuming peak load

Considering our assumption of using daily active users as a proxy for the number of requests per second to find the number of servers for peak load times, we get 500 million requests per second. Then, we use the following formula to calculate the number of servers:

$$Servers\ needed\ at\ peak\ load = \frac{Number\ of\ requests/second}{RPS\ of\ server}$$

$$Servers\ needed\ at\ peak\ load = \frac{500\ million}{64,000} = 7812.5 \approx 8K\ servers$$



Number of servers required for the Instagram system under peak load assumption

> **Note:** Concurrent requests significantly impact the number of required servers compared to requests spread over time. For a more in-depth exploration of refining peak load assumptions, refer to the "Back-of-the-envelope Calculations" chapter.

Try it yourself

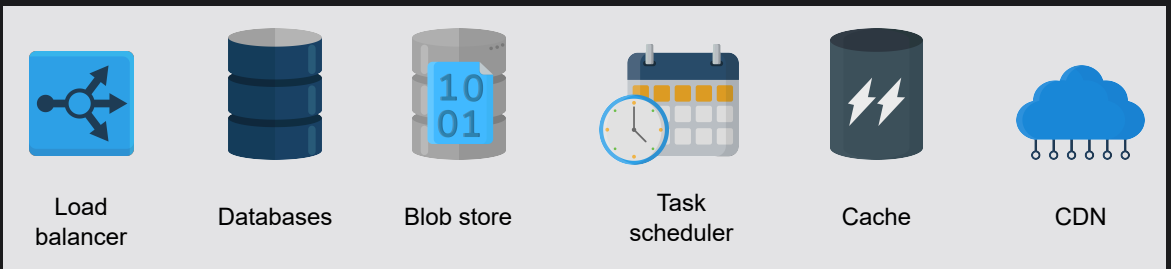
Let's analyze how the number of photos per day affects the storage and bandwidth requirements. For this purpose, try to change values in the following table to compute the estimates. We have set 20 requests by each user per day in the following calculation:

Requirements	Calculations
Number of photos per day (in millions)	60

Size of each photo (in MB)	3
Number of videos per day (in millions)	35
Size of each video (in MB)	150
Number of daily active users (in millions)	500
Number of requests a server can handle per day	691200000
Storage estimation per day (in TB)	*f* 5430
Incoming bandwidth (Gb/s)	*f* 502.8
Number of servers needed	*f* 15

Building blocks we will use

In the next lesson, we'll focus on the high-level design of Instagram. The design will utilize many building blocks that have been discussed in the initial chapters also. We'll use the following building blocks in our design:



Building blocks used in Instagram design

- A **load balancer** at various layers will ensure smooth requests distribution among available servers.

- A **database** is used to store the user and accounts metadata and relationship among them.
- **Blob storage** is needed to store the various types of content such as photos, videos, and so on.
- A **task scheduler** schedules the events on the database such as removing the entries whose time to live exceeds the limit.
- A **cache** stores the most frequent content related requests.
- **CDN** is used to effectively deliver content to end users which reduces delay and burden on end-servers.

In the next lesson, we'll discuss the high-level design of the Instagram system.