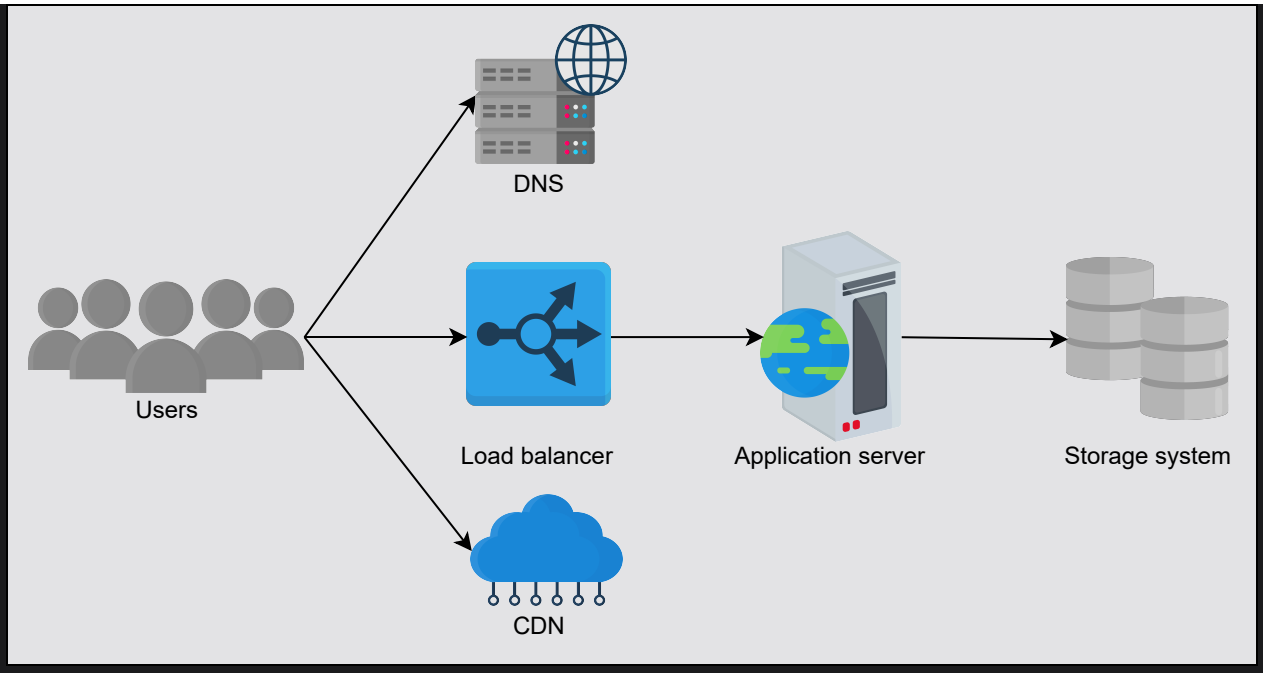
High-level Design of Twitter

Understand the high-level design of the Twitter service.

We'll cover the following

User-system interaction

Let’s begin with the high-level design of our Twitter system. We’ll initially highlight and discuss the building blocks, as well as other components, in the context of the Twitter problem briefly. Later on, we’ll dive deep into a few components in this chapter.



Twitter components

- **Users** post Tweets delivered to the server through the load balancer. Then, the system stores it in persistent storage.
- **DNS** provides the specified IP address to the end user to start communication with the requested service.
- **CDN** is situated near the users to provide requested data with low latency. When users search for a specified term or tag, the system first searches in the CDN proxy servers containing the most frequently requested content.
- **Load balancer** chooses the operational application server based on traffic load on the available servers and the user requests.
- **Storage system** represents the various types of storage (SQL-based and NoSQL-based) in the above illustration. We'll discuss significant storage systems later in this chapter.
- **Application servers** provide various services and have business logic to orchestrate between different components to meet our functional requirements.

We have detailed chapters on DNS, CDN, specified storage systems (Database Key-value store, Blob store), and Load balancers in our building blocks section.

We'll focus on further details specific to the Twitter service in the coming lessons. Let's first understand the service API.

API design

This section will focus on designing various APIs regarding the functionalities we are providing. We learn how users request various services through APIs. We'll only concentrate on the significant parameters of the APIs that are relevant to our design. Although the front-end server can call another API or add more parameters in the API received from the end users to fulfill the given request, we consider all relevant arguments specified for the particular request in a single API. Let's develop APIs for each of the following features:

- Post Tweet
- Like or dislike Tweet
- Reply to Tweet
- Search Tweet
- View user or home timeline
- Follow or unfollow the account
- Retweet a Tweet

Post Tweet

The POST method is used to send the Tweet to the server from the user through the `/postTweet` API.

```
postTweet(user_id, access_type, tweet_type, content, tweet_length, media_field, po
st_time, tweet_location, list_of_used_hashtags, list_of_tagged_people)
```

Let's discuss a few of the parameters:

Parameter	Description

`user_id`	It indicates the unique ID of the user who posted the Twee
`access_type`	It tells us whether the Tweet is protected (that is, only visible to followe
`tweet_type`	It indicates whether the Tweet is text-based, video-clip based, image(consisting of different types.
`content`	It specifies the Tweet's actual content (text).
`tweet_length`	It represents the text length in the Tweet. In the case of video, it tells us t size of a video.
`media_field`	It specifies the type of media (image, video, GIF, and so on) delivered i

The rest of the parameters are self-explanatory.

> **Note:** Twitter uses the **Snowflake** service to generate unique IDs for Tweets. We have a detailed chapter (Sequencer) that explains this service.

Points to Ponder

Question 1

At most, how many hashtags can a Tweet have?

Show Answer

Like or dislike Tweet

The `/likeTweet` API is used when users like public Tweets.

```
likeTweet(user_id, tweet_id, tweeted_user_id, user_location)
```

Parameter	Description
`user_id`	It indicates the unique ID of the user who liked the Twee
`tweet_id`	It indicates the Tweet's unique ID.
`tweeted_user_id`	This is the unique ID of the user who posted the Tweet
`user_location`	It denotes the location of the user who liked the Tweet

The parameters above are also used in the `/dislikeTweet` API when users dislike others' Tweets.

Reply to Tweet

The `/replyTweet` API is used when users reply to public Tweets.

```
replyTweet(user_id, tweet_id, tweeted_user_id, reply_type, reply_content, reply_le
ngth)
```

The `reply_type`, `reply_content`, and `reply_length` parameters are the same as `tweet_type`, `content`, and `tweet_length` respectively.

Search Tweet

When the user searches any keyword in the home timeline, the GET method is used. The following is the `/searchTweet` API:

```
searchTweet(user_id, search_term, max_result, exclude, media_field, expansions, so
rt_order, next_token, user_location)
```

Some new parameters introduced in this case are:

Parameter	Description
`search_term`	It is a string containing the search keyword or phrase.
`max_result`	It is the number of Tweets returned per response page. By default, the Tv is 10.
`exclude`	It specifies what to exclude from the returned Tweets, that is, replies a maximum limit on returned Tweets is 3200, but when we exclude replies, is reduced to 800 Tweets.
`media_field`	It specifies the media (image, video, GIF) delivered in each retur
`expansions`	It enables us to request additional data objects in the returned Tweet mentioned user, referenced Tweet, attached media, attached places ob
`sort_order`	It specifies the order in which Tweets are returned. By default, it will retu Tweets first.
`next_token`	It is used to get the next page of results. For instance, if `max_result` is s and the result set contains 200 Tweets, then the value of `next_token` is c the response to request the next page containing the following 100 Twe (page) will not have a `next_token`.

Response

Let's look at a sample response in JSON format. The **id** is the user's unique ID who posted the Tweet and the **text** is the Tweet's content. The **result_count** is t count of the returned Tweet, which we set in the `max_result` in the request. Here, we're displaying the default fields only.

Click to see response in JSON

> **Note:** Twitter performs various types of searches. The following are two of them:
>
> - One search type returns the result of the last seven days, which all registered users usually use.
> - The other type returns all matching results on all Tweets ever posted (remind that service does not delete a posted Tweet). Indeed, matches can contain the first Tweet on Twitter. This search is usually used for academic research.

educative

View home_timeline

The GET method is suitable when users view their home timelines through the `/viewHome_timeline` API.

```
viewHome_timeline(user_id, tweets_count, max_result, exclude, next_token, user_location)
```

In the `/viewHome_timeline` API, we'll exclude the `user_location` to get the user timeline.

The `max_result` parameter determines the number of tweets a client application can show the user. The server sends the `max_result` number of tweets in each response. Further, the server will also send a paginated `list_of_followers` to reduce the client latency.

Point to Ponder

Question

Which parameter in the `viewHome_timeline` method is the most relevant when deciding which promoted ads (Tweets) to be returned in response?

Show Answer

Follow the account

The `/followAccount` API is used when users follow someone's account on Twitter.

```
followAccount(account_id, followed_account_id)
```

Parameter	Description
`account_id`	It specifies the unique ID of a user who follows that account or
`followed_account_id`	It indicates the unique ID of the account that the user follo

The `/unfollowAccount` API will use the same parameters when a user unfollows someone's account on Twitter.

Retweet a Tweet

When a registered user Retweets (re-posts) someone's Tweet on Twitter, the following `/retweet` API is called:

```
retweet(user_id, tweet_id, retweet_user_id)
```

The same parameters will be required in the `/undoRetweet` API when users undo a Retweet of someone's Tweet.