**DESIGN A Chat System**

News feed is the constantly updating list of stories in the middle of your home page. News Feed includes status updates, photos, videos, links, app activity, and likes from people, pages, and groups that you follow on Facebook

### Understanding Problem and Establishing Design Scope:

Query1: What type of feeds will be supported? Text, Image, video?

-> It can contain media files, including both images and videos.

Query2: Is this a mobile app? Web app? Or both?

-> Both

Query3: What is the total number of active users?

-> 10 million DAU

Query4: How many friends can a user have?

-> 5000

**Query5: Is the news feed sorted by reverse chronological order or any [articular order such as topic scores? For instance, posts from your close friends have higher scores.**

**-> To keep things simple, let us assume the feed is sorted by reverse chronological order.**

Query6: What are the important features?

-> users can publish a post and see her friend’s posts on the new feed.

### Back of the Envelope Estimation:

### High Level Design Propositions and approaches:

The design is divided into two flows:

1. Feed publishing: When a user publishes a post, corresponding data is written into cache and database. A post is populated to her friend’s news feed.
2. Newsfeed building: For simplicity, let us assume the news feed is built by aggregating friend’s posts in reverse chronological order.

**Newsfeed APIs**

Two most important APIs:

1. Feed publishing API: To publish a post, a HTTP POST request will be sent to the server. The API is shown below:  
   POST /v1/me/feed

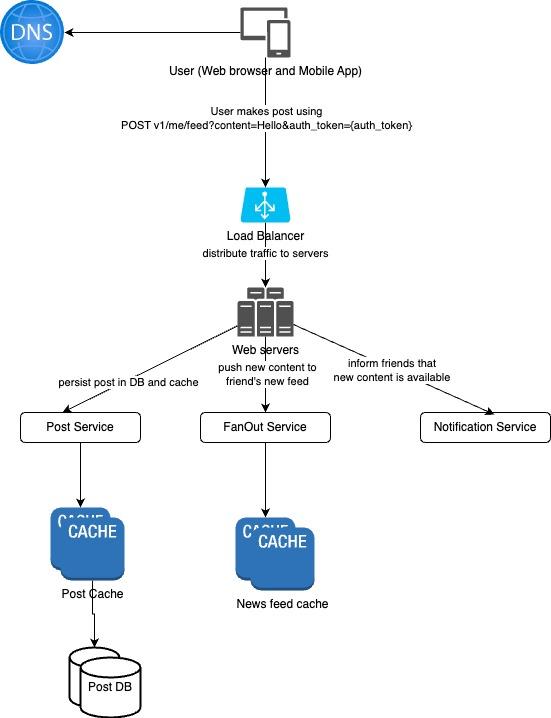
Params:

* Content: content is the text of the post
* Auth\_token: it is used to authenticate API requests.

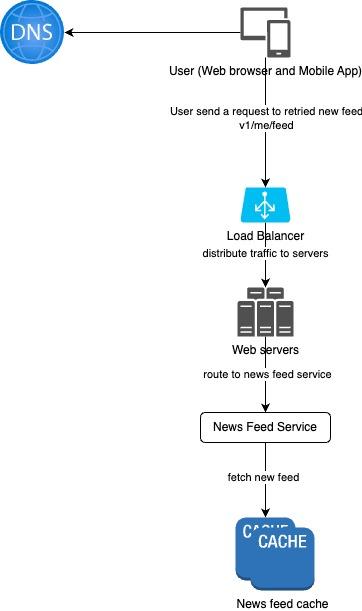
1. Newsfeed retrieval API:  
   The API to retrieve news is shown below:  
   GET /v1/me/feed

Params:  
- auth\_token: It is used to authenticate API requests.

**Feed publishing**

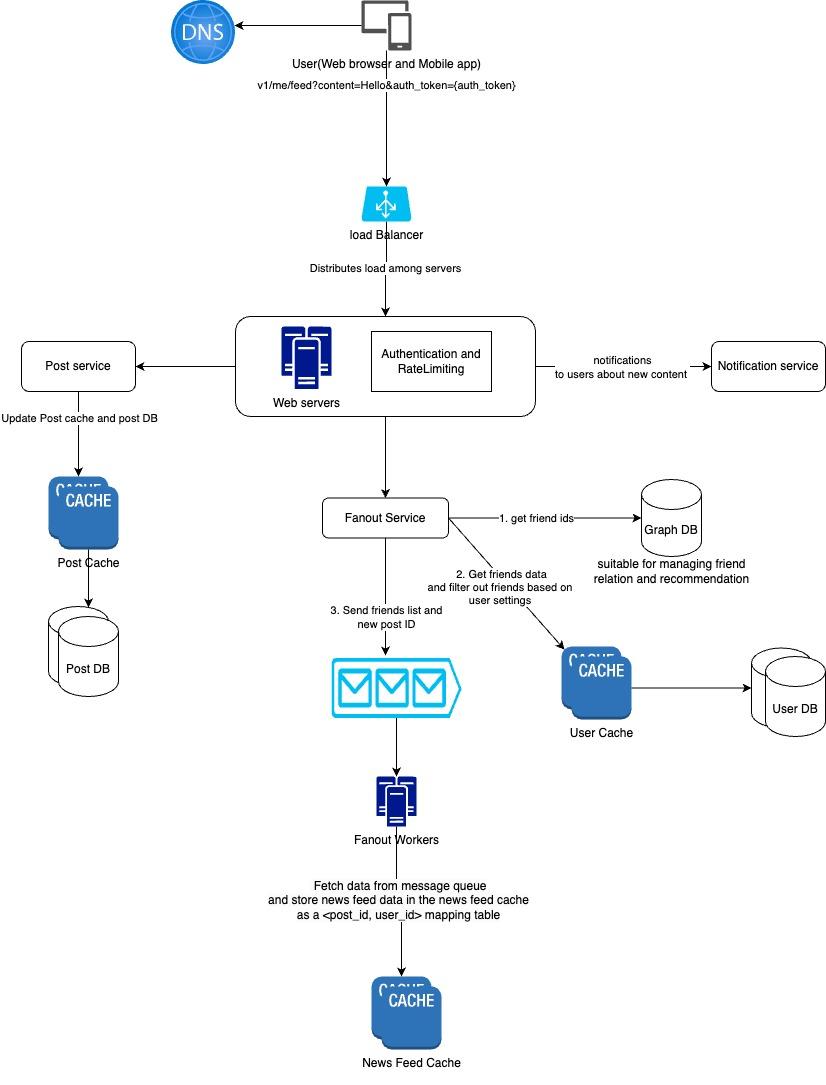
****

**NewsFeed building**

****

### Design Deep Dive:

**Feed Publishing deep dive**

**  
Web servers:**

Besides communicating with clients, web servers enforce authentication and rate-limiting.

Only users signed in with valid auth\_token are allowed to make posts. The system limits the number of posts a user can make within a certain period, vital to prevent spam and abusive content.

**Fan out service:**

Fanout is the process of delivering a post to all friends.

Two type of fanout models are:  
1. Fanout on write(also call push model)

News feed is pre-computed during write time. A new post is delivered to a friend's cache immediately after it is published.

Pros:

* The news feed is generated in real-time and can be pushed to friends immediately
* Fetching news feed is fast because new feed is pre-computed during write time.

Cons:

* If a user has many friends, fetching friends list and generating news feeds for all of them are slow and time consuming. **It is called the hotkey problem.**
* For inactive users or those who rarely log in, pre-computing news feeds waste computing resources.

2. Fanout on read (also called pull module)

The news feed is generated during read time. Recent posts are pulled when a user loads her home page.

Pros:

• For inactive users or those who rarely log in, fanout on read works better because it will not waste computing resources on them.

• Data is not pushed to friends so there is no hotkey problem.

Cons:  
 • Fetching the news feed is slow as the news feed is not pre-computed.

**We adopt a hybrid approach.**Since fetching news feeds fast is crucial, we use a push model for the majority of users.

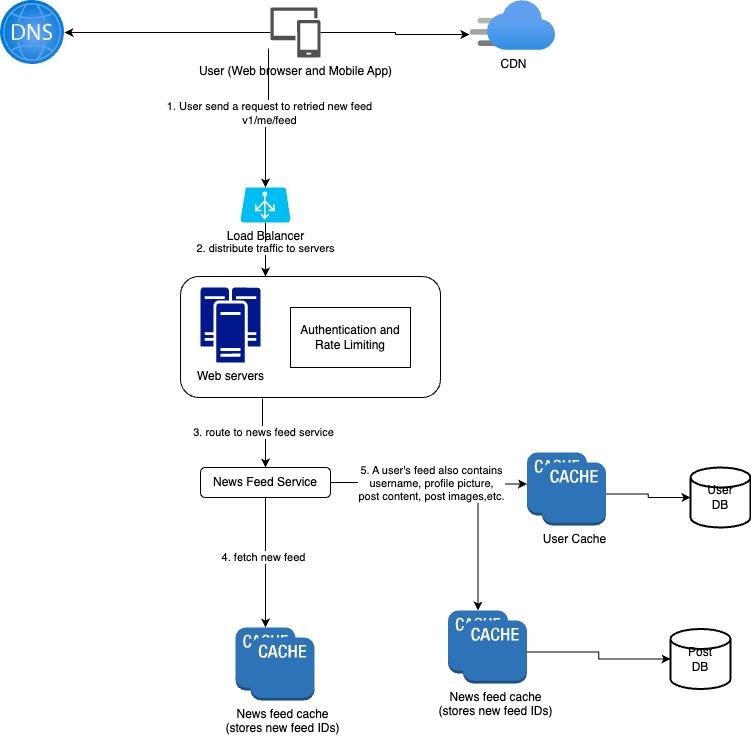
For celebrities or users who have many friends/followers, we let followers pull news content on-demand to avoid overload.

Consistent hashing is a useful technique to mitigate the hotkey problem as it helps to distribute requests/data more evenly.

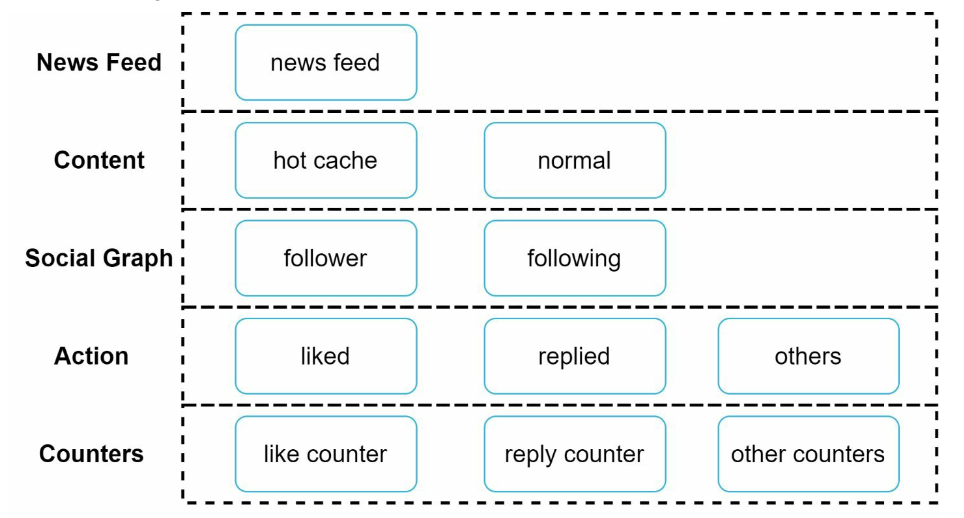
The **fanout service works** as follows:

1. Fetch friend IDs from graph database. Graph databases are suited for managing friend relationships and friend recommendations.
2. Get friends info from the user cache. The system then filters out friends based on user settings. For example, if you mute someone, her posts will not show up on your news feed even though you are still friends. Another reason why posts may not show is that a user could selectively share information with specific friends or hide it from other people.
3. Send friends list and new post ID to the message queue.
4. Fanout workers fetch data from the message queue and store news feed data in the news feed cache. You can think of the news feed cache as a <post\_id, user\_id> mapping table.
5. The chance of a user scrolling through thousands of posts in the news feed is slim. Most users are only interested in the latest content, so the cache miss rate is low.
6. Store <post\_id, user\_id > in the news feed cache.

**Neesfeed retrieval deep dive**



**Cache Architecture**



News Feed stores IDs of news feed

Content stores every post data. Popular content is stored in hot cache

Social Graph: Stores user relationship data

Action : Stores info about whether a user liked a post, replied to a post, or took other actions on a post.

Counters: stores counters for like, reply, follower, following, etc.

### Additional optimization:

Scaling the database:  
 • Vertical scaling vs Horizontal scaling • SQL vs NoSQL  
 • Master-slave replication  
 • Read replicas  
 • Consistency models  
 • Database sharding

Other talking points:

• Keep web tier stateless

• Cache data as much as you can

• Support multiple data centers

• Loose couple components with message queues

• Monitor key metrics. For instance, QPS during peak hours and latency while users refreshing their news feed are interesting to monitor.