**Chapter 11**

Design a News Feed System

Newsfeed – constantly updating list of stories in the middle of your home page

* Can include status updates, photos, videos, links,
* Has app activity such as likes, comments from people, pages, and groups
* Popular interview question

*Questions*

* Is this a mobile app? Web app or both?
* What are the most important features?
* How should newsfeed articles be sorted? e.g. reverse chronologically, by topic or interest scores, by interaction scores with other users such as a close friend’s newsfeed may be prioritized?
* Is there a limit to how many friends a user can have?
* How many daily users do we expect to support?
* Can a feed contain images, videos, or just text?

*High level designs*

*Feed publishing* – when a user publishes a post, data is written into cache and database

* Newsfeed publishing API
  + POST request
  + Send contents to the servers to be cached or stored
  + Pass an authentication token for validation and identify the user

*Newsfeed building* – the method of which the newsfeed is generated, given what is prioritized for sorting and data retrieval

* Newsfeed retrieval API
  + GET request
  + Pass an authentication token for retrieve data

*Feed publishing*

A diagram of a software flow

Description automatically generated

*Web servers*

* Authentication – only user signed with valid auth tokens are allowed to make posts
* Rate limiting – limits the number of posts a user can make at a given time to reduce spam and prevent abusive content

Fanout services – process of delivering a post to all friends

*Fanout on write (Push model) – n*ewsfeed is pre-computed during write time

* Pros
  + Newsfeed is generated in real-time and can be pushed to friends immediately
  + Fetching newsfeed is fast because it is pre-computed during write time
* Cons
  + Hotkey problem – if a user has many friends, fetching the friend list and generating news feeds for all of them is slow and time consuming
  + For inactive users, pre-computing is a waste of computing resources

*Fanout on read (Pull model)* – newsfeed is generated during read time

* Pros
  + Resources are allocated correctly based on how active a user is
  + No hotkey problem since data is not constantly pushed to friends
* Cons
  + Fetching newsfeed is slow

*Hybrid approach*

* Use push model for majority of active users
* Use pull model for celebrities or user with many friends/followers
* Consistent hashing helps mitigate the hotkey problem, allowing data to be distributed more evenly

*Fanout service workflow*

1. User sends a request to post new content
2. A load balancer distributes requests to web servers
   * A server to post the user’s contents
   * A server to update the notification service
   * A server to fanout service to update the newsfeed of the user’s friends
3. Fanout service will fetch friend IDs from graph database
   * Graph database – suited for managing friend relationship and friend recommendations
4. Get friends info from the user cache
   * Filters out friends based on user settings (e.g. muting or ignoring)
5. Send friends list and new post ID to the message queue
6. Fanout worker fetch data from the message queue and store them in the cache
   * E.g. a newsfeed cache can look like (*post\_id*, *user\_id*) mapping table
   * Only IDs are stored in the cache to protect memory
   * A configurable limit can be set in the cache – only storing the latest content rather than everything they missed in between logins

*Newsfeed retrieval*

A diagram of a software development

Description automatically generated

*Newsfeed retrieval workflow*

1. A user sends a request to retrieve their newsfeed
2. A load balancer redistributes requests to web servers
3. Web servers call the newsfeed service to fetch newsfeeds
4. Newsfeed service gets a list post IDs from the newsfeed cache
5. Newsfeed service fetches the complete user and post objects from user and post caches to construct the fully hydrated news feed
   * Fully hydrated news feed – contains different tiers of data such as username, profile pictures, post objects, etc.
6. Fully hydrated newsfeed is returned in JSON format to the client and rendered

*Cache architecture*

* Can be divided into multiple layers, such as

A group of rectangular objects with black text

Description automatically generated

* + *Newsfeed* – stores IDs of new feeds
  + *Content* – stores every post data.
    - Important content such as popular content is stored in hot cache
  + *Social* *graph* – stores user relationship data
  + *Action* – stores info related to user interactions on posts
    - User likes and comments
    - User tagging original poster or other users
    - The number of times a post is shared and saved
  + *Counters* – stores like, reply, followers, and following, etc.