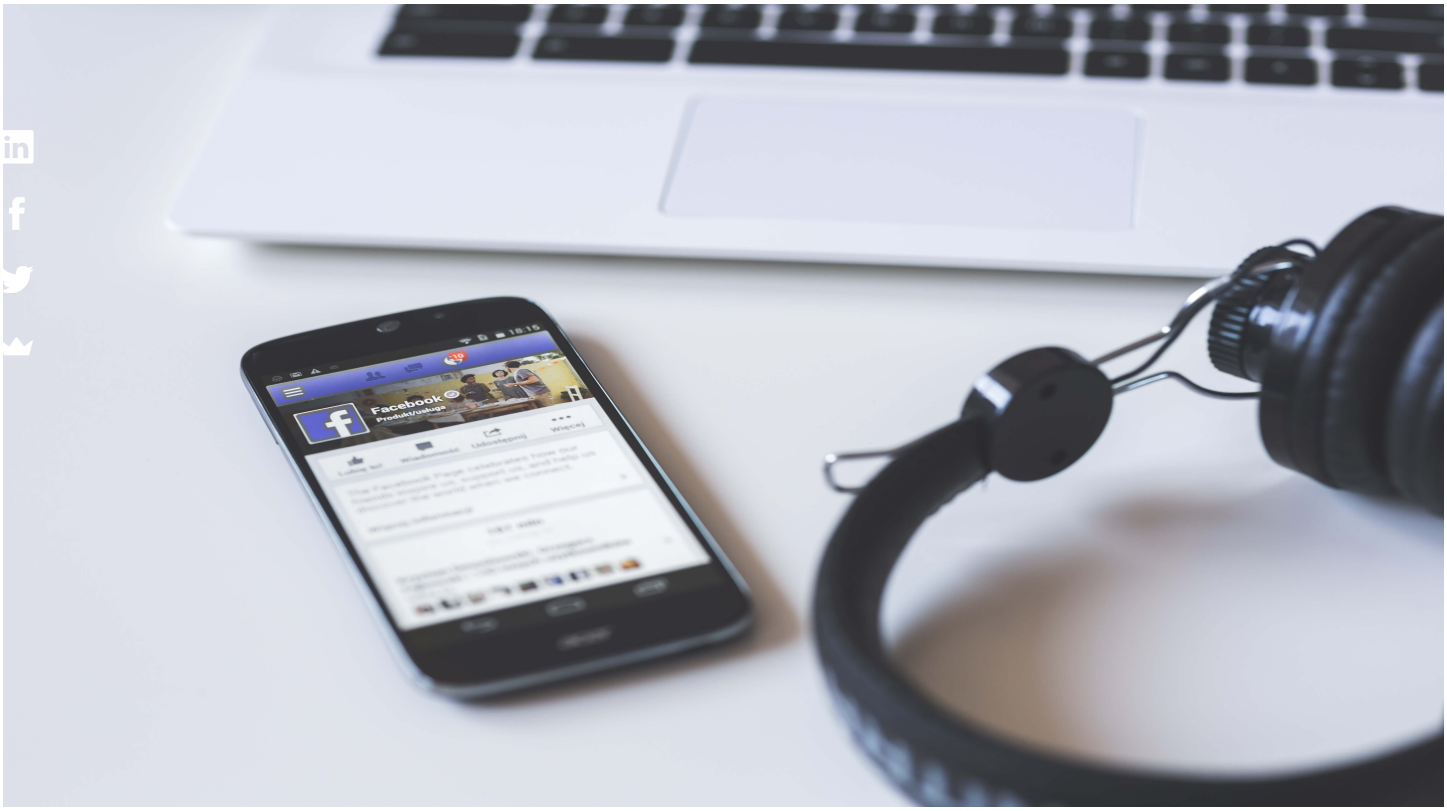

Design News Feed System (Part 2)

This is the 2nd part for system design interview question analysis – design news feed system. If you haven't seen our first post (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/), please go check it.

Just briefly summarize what we have discussed in part 1. We started with a simple question – how to design news feed system for Facebook that allows users see feeds/updates from friends. We modeled the whole system using relational database (https://en.wikipedia.org/wiki/Relational_database) and talked about pros and cons of different design.

Ranking is an interesting topic for news feed system. We explained some general ideas of ranking in previous post. And in this post, we'll continue the discussion ranking and also cover topics like feed publishing and so on.

Ranking – continue

The general idea of ranking is to first select features/signals that are relevant and then figure out how to combine them to calculate a final score. This approach is extremely common among lots of real-world systems.

As you can see that what matters here are two things – **features** and **calculation algorithm**. To give you a better idea of it, I'd like to briefly introduce how ranking actually works at Facebook – EdgeRank (https://en.wikipedia.org/wiki/EdgeRank).

For each news update you have, whenever another user interacts with that feed, they're creating what Facebook calls an Edge, which includes actions like like and comments.

First of all, let's take a look at what features are used to evaluate the importance of an update/feed. Edge Rank basically is using three signals: affinity score, edge weight and time decay.

- Affinity score (u). For each news feed, affinity score evaluates how close you are with this user. For instance, you are more likely to care about feed from your close friends instead of someone you just met once. You might ask how affinity score is calculated, I'll talk about it soon.
- Edge weight (e). Edge weight basically reflects importance of each edge. For instance, comments are worth more than likes.
- Time decay (d). The older the story, the less likely users find it interesting.

So how does Facebook rank feeds by these three features? The calculation algorithm is quite straightforward. For each feed you create, multiply these factors for each Edge then add the Edge scores up and you have an update's EdgeRank. And the higher that is, the more likely your update is to appear in the user's feed.

Affinity score

We can do exactly the same thing to evaluate affinity score.

Various factors can be used to reflect how close two people are. First of all, explicit interactions like comment, like, tag, share, click etc. are strong signals we should use. Apparently, each type of interaction should have different weight. For instance, comments should be worth much more than likes.

Secondly, we should also track the time factor. Perhaps you used to interact with a friend quite a lot, but less frequent recently. In this case, we should lower the affinity score. So for each interaction, we should also put the time decay factor.

To sum up the ranking section, I hope this common approach for ranking can be one of your takeaways. Also, EdgeRank was first published at 2010 and it can be outdated.

Feed publishing

When a user loads all the feeds from his friends, it can be an extremely costly action. Remember that a user can have thousands of friends and each of them can publish a huge amount of updates especially for high profile users. To load all feeds from friends, the system requires at least two joins (get friends list and feed list.

So how to optimize and scale the feed publishing system?

Basically there are two common approaches here – **push** and **pull**.

For a push system, once a user has published a feed, we immediately pushing this feed (actually the pointer to the feed) to all his friends. The advantage is that when fetching feed, you don't need to go through your friends list and get feeds for each of them. It significantly reduces read operation. However, the downside is also obvious. It increases write operation especially for people with a large number of friends.

For a pull system, feeds are only fetched when users are loading their home pages. So feed data doesn't need to be sent right after it's created. You can see that this approach optimizes for write operation, but can be quite slow to fetch data even after using denormalization (https://en.wikipedia.org/wiki/Denormalization) (check our previous post (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/) if you don't understand this).

Both approaches work well at certain circumstances and it's always better to understand their pros and cons.

Selective fanout

The process of pushing an activity to all your friends or followers is called a fanout. So the push approach is also called fanout on write, while the pull approach is fanout on load.

Here I'd like to ask if you have any approaches to further optimize the fanout process?

In fact, you can do a combination of both. Specifically, if you are mainly using push model, what you can do is to disable fanout for high profile users and other people can only load their updates during read. The idea is that push operation can be extremely costly for high profile users since they have a lot of friends to notify. By disabling fanout for them, we can save a huge number of resources. Actually Twitter has seen great improvement after adopting this approach.

By the same token, once a user publish a feed, we can also limit the fanout to only his active friends. For non-active users, most of the time the push operation is a waste since they will never come back consuming feeds.

Summary

If you follow 80-20 rule (https://en.wikipedia.org/wiki/Pareto_principle), 80% of the cost comes from 20% of features/users. As a result, optimization is really about identifying the bottleneck.

Also, feed system is a very popular topics since it's widely used by so many products nowadays. If you are interested in this topic and want to explore more, I'd recommend you take a look at the following resources:

- Yahoo's New Research Model (https://yahooresearch.tumblr.com/post/139436148571/yahoos-new-research-model)
- Etsy Activity Feeds Architecture (http://www.slideshare.net/danmckinley/etsy-activity-feeds-architecture/)
- Twitter's Approach (http://www.slideshare.net/nkallen/q-con-3770885)

Related Posts:

1. **Design News Feed System (Part 1) (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/)**
2. **How to Design Twitter (Part 2) (http://blog.gainlo.co/index.php/2016/02/24/system-design-interview-question-how-to-design-twitter-part-2/)**
3. **Design Facebook Chat Function (http://blog.gainlo.co/index.php/2016/04/19/design-facebook-chat-function/)**
4. **How to Design Twitter (Part 1) (http://blog.gainlo.co/index.php/2016/02/17/system-design-interview-question-how-to-design-twitter-part-1/)**

One thought on "Design News Feed System (Part 2)"

ASWIN
February 13, 2017 at 6:28 am (http://blog.gainlo.co/index.php/2016/04/05/design-news-feed-system-part-2/#comment-5476)

Thanks a lot for the article. Great explanation on the ranking, push and pull fanout.

Reply (http://blog.gainlo.co/index.php/2016/04/05/design-news-feed-system-part-2/?replytocom=5476#respond)

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

5 Things To Do After You Failed an Interview (http://blog.gainlo.co/index.php/2016/04/01/5-things-to-do-after-you-failed-an-interview/)

If a String Contains an Anagram of Another String (http://blog.gainlo.co/index.php/2016/04/08/if-a-string-contains-an-anagram-of-another-string/)

CATEGORIES

Coding Interview Questions (http://blog.gainlo.co/index.php/category/coding-interview-questions/)

Common Pitfalls (http://blog.gainlo.co/index.php/category/common-pitfalls/)

Common Questions (http://blog.gainlo.co/index.php/category/common-questions/)

Facebook Interview Questions (http://blog.gainlo.co/index.php/category/facebook-interview-questions/)

Mock Interview Thoughts (http://blog.gainlo.co/index.php/category/mock-interview-thoughts/)

Others (http://blog.gainlo.co/index.php/category/others/)

System Design Interview Questions (http://blog.gainlo.co/index.php/category/system-design-interview-questions/)

The Complete Guide to Google Interview Preparation (http://blog.gainlo.co/index.php/category/google-interview-preparation/)

Uber Interview Questions (http://blog.gainlo.co/index.php/category/uber-interview-questions/)