System Design: The Sharded Counters

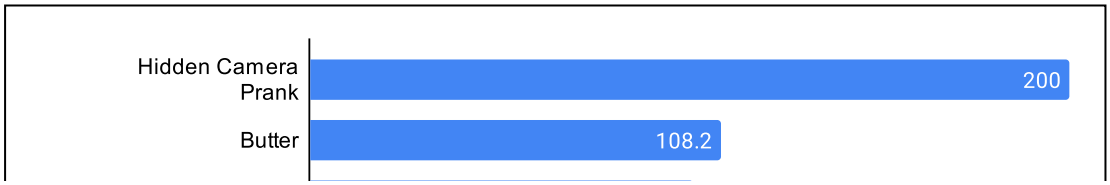
Get introduced to sharded counters.

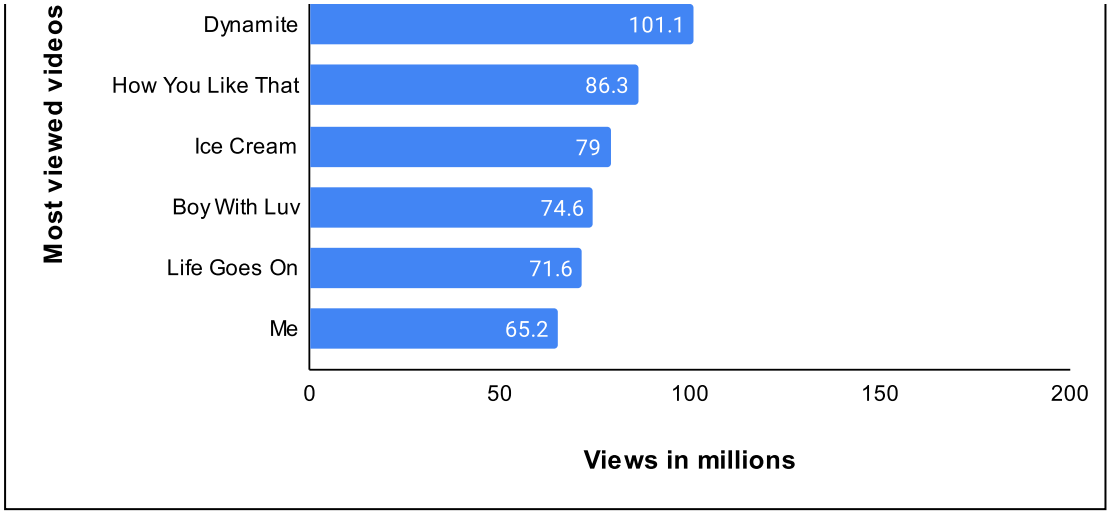
We'll cover the following

Problem statement

Real-time applications like Facebook, Twitter, and YouTube have high user traffic. Users interact with the applications and perform multiple operations (view, like, comment, and so on) depending on the application's structure. For instance, an image is posted on a Facebook page that has millions of followers, and the post's likes rapidly increase after each millisecond. Here, it might be easy to count the likes for this single image, but what will we do when thousands of such images or videos are uploaded simultaneously by many celebrities, each with millions of followers. This problem is known as the **heavy hitters problem**.

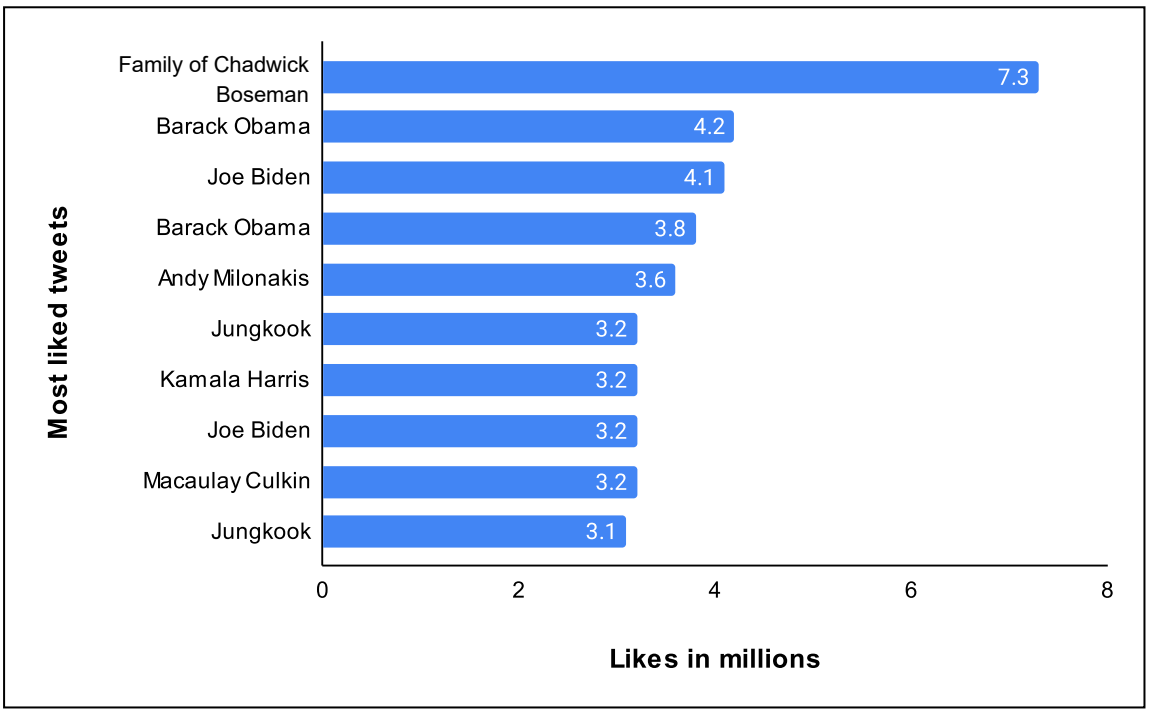
The above scenario shows how a simple counting operation becomes challenging to manage with precision and performance. The following figure shows YouTube's videos that were viewed by millions of users in a 24-hour span in August 2021:





YouTube videos' views in 24 hours

On average, six thousand tweets are sent on Twitter within one second, which equals 360,000 tweets per minute and about 500 million tweets per day. A challenging task is to handle billions of likes on these 500 million tweets per day. The following table shows the most liked tweets in one day as of 2022:



Most liked tweets in a single day

How will we handle millions of write requests coming against the likes on thousands of tweets per minute? The challenge is that writing takes more

time than reading, and concurrent activity makes this problem harder. As the number of concurrent writes increases for some counter (which might be a variable residing in a node's memory), the lock contention increases non-linearly. After some point, we might spend most of the time acquiring the lock so that we could safely update the counter.

How will we design sharded counters?

We have divided the design of sharded counters into three lessons:

1. **High-level Design**: We'll discuss the high-level design of sharded counters in this lesson. In addition, we'll also briefly explain the API design.
2. **Detailed Design**: This lesson will dive deeply into the design of sharded counters. Moreover, we'll also evaluate our proposed design.
3. **Quiz**: We'll review major concepts of sharded counters design with a quiz.

Let's begin with the high-level solution sketch of sharded counters.

Back

Evaluation of a Distributed Task Scheduler's ...

Next

High-level Design of Sharded Counters

Mark as Completed