Dropbox Interview – Design Hit Counter

It starts with a simple question – if you are building a website, how do you count the number of visitors for the past 1 minute?

"Design hit counter" problem has recently been asked by many companies including Dropbox and the question is harder than it seems to be. This week, we'll uncover all the mysteries of the problem. A couple of topics are discussed including basic data structures design, various optimization, concurrency and distributed counter.

What's special about this problem?

I always like to tell our readers why we select this question to analyze so that you'll know exactly whether it's worth your time to read. As an interviewer, I have a strong preference for questions that are not hard to solve in the simplest case but the discussion can go deeper and deeper by removing/adding specific conditions. And this question is exactly the case.

Also, the question doesn't come from nowhere, but has real use cases. For many systems today, we need a system to track not only users numbers, but different types of request numbers in real time.

If you haven't thought about this problem, spend some time working on it before reading following sections.

Simple case

Forget about all the hard problems like concurrency and scalability issue, let's say we only have a single machine with no concurrent requests, how would you get the number of visitors for the past 1 minute?

Apparently, the simplest solution is to store all the visitors with the timestamps in the database. When someone asks for visitor number of the past minute, we just go over the database and do the filtering and counting. A little bit optimization is to order users by timestamp so that we won't scan the whole table.

The solution is not efficient as the time complexity is O(N) where N is the number of visitors. If the website has a large volume, the function won't be able to return the number immediately.

Let's optimize

A couple of ways to think about this problem. Since the above approach returns not only visitor numbers, but also visitors for the past minute, which is not needed in the question. And this is something we can optimize. From a different angle, we only need numbers for the past minute instead of any time range, which is another area that we can improve potentially. In a nutshell, by removing unnecessary features, we can optimize our solution.

A straightforward idea is to only keep users from the past minute and as time passes by, we keep updating the list and its length. This allows us to get the number instantly. In essence, we reduce the cost of fetching the numbers, but have to keep updating the list.

We can use a queue or linked list to store only users from the past minute. We keep all the element in order and when the last user (the earliest user) has the time more than a minute, just remove it from the list and update the length.

Space optimization

There's little room to improve the speed as we can return the visitor number in O(1) time. However, storing all the users from the past minute can be costly in terms of space. A simple optimization is to only keep the user timestamp in the list rather than the user object, which can save a lot of space especially when the user object is large.

If we want to further reduce the space usage, what approach would you take?

A good way to think about this is that to improve space complexity, what should we sacrifice? Since we still want to keep the time complexity O(1), one thing we can compromise is accuracy. If we can't guarantee to return the most accurate number, can we use less space?

Instead of tracking users from the past minute, we can only track users from the past second. By doing this, we know exactly how many visitors are from the last second. To get visitor numbers for the past minute, we keep a queue/linked list of 60 spots representing the past 60 seconds. Each spot stores the visitor number of that second. So every second, we remove the last (the earliest) spot from the list and add a new one with the visitor number of past second. Visitor number of the past minute is the sum of the 60 spots.

The minute count can be off by the request of the past second. And you can control the trade-off between accuracy and space by adjusting the unit, e.g. you can store users from past 2 seconds and have 30 spots in the list.

How about concurrent requests?

In production systems, concurrency is the most common problems people face. If there can be multiple users visiting the site simultaneously, does the previous approach still work?

Part of. Apparently, the basic idea still holds. However, when two requests update the list simultaneously, there can be race conditions (https://en.wikipedia.org/wiki/Race_condition). It's possible that the request that updated the list first may not be included eventually.

The most common solution is to use a lock to protect the list. Whenever someone wants to update the list (by either adding new elements or removing the tail), a lock will be placed on the container. After the operation finishes, the list will be unlocked.

This works pretty well when you don't have a large volume of requests or performance is not a concern. Placing a lock can be costly at some times and when there are too many concurrent requests, the lock may potentially block the system and becomes the performance bottleneck.

Distribute the counter

When a single machine gets too many traffic and performance becomes an issue, it's the perfect time to think of distributed solution. Distributed system (https://en.wikipedia.org/wiki/Distributed_computing) significantly reduces the burden of a single machine by scaling the system to multiple nodes, but at the same time adding complexity.

Let's say we distribute visit requests to multiple machines equally. I'd like to emphasize the importance of equal distribution first. If particular machines get much more traffic than the rest machines, the system doesn't get to its full usage and it's very important to take this into consideration when designing the system. In our case, we can get a hash of users email and distribute by the hash (it's not a good idea to use email directly as some letter may appear much more frequent than the others).

To count the number, each machine works independently to count its own users from the past minute. When we request the global number, we just need to add all counters together.

Summary

One of the reasons I like this question is that the simplest solution can be a coding question and to solve concurrency and scalability issue, it becomes a system design question. Also, the question itself has a wide usage in production systems.

Again, the solution itself is not the most important thing in the post. What we're focused on is to illustrate how to analyze the problem. for instance, trade-off is a great concept to be familiar with and when we try to optimize one area, think about what else should be sacrificed. By thinking like this, it opens up a lot of doors for you.

By the way, if you want to have more guidance from experienced interviewers, you can check Gainlo (http://www.gainlo.co/?utm_source=blog&utm_medium=Dropbox%20Interview%20-%20Design%20Hit%20Counter&utm_campaign=blog) that allows you to have mock interview with engineers from Google, Facebook etc..

Related Posts:

1. **Design Facebook Chat Function (http://blog.gainlo.co/index.php/2016/04/19/design-facebook-chat-function/)**
2. **Design a Cache System (http://blog.gainlo.co/index.php/2016/05/17/design-a-cache-system/)**
3. **How to Design a Trending Algorithm for Twitter (http://blog.gainlo.co/index.php/2016/05/03/how-to-design-a-trending-algorithm-for-twitter/)**
4. **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/)**

---

8 thoughts on "Dropbox Interview – Design Hit Counter"

BRAJ
November 21, 2016 at 6:57 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-3610)

System concurrency and load balancing feature is less explanatory.

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=3610#respond)

AKASH MUTHA
December 9, 2016 at 1:58 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-3914)

Hey, How is the actual time complexity O(1). Why are we not considering the time for updating the list?

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=3914#respond)

ADAM
January 3, 2017 at 6:57 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-4484)

Removing the head is the O(1) or at worst O(60) ~> O(1) for Linked List or a Queue unless I'm mistaken

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=4484#respond)

BRUCE
January 12, 2017 at 4:45 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-4683)

Why wouldn't you just recognize that this is a metric you want to keep in your metric system, update with timestamp for all servers and provide access to the value as past minute, hour, day, etc from the telemetry/metric system? Worse comes to worse, you log access and issue a splunk (or favorite bigdata log consumer) query.

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=4683#respond)

ALI
February 15, 2017 at 2:32 pm (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-5523)

This can be a lot trickier if you want to count number of unique visitors. I.e. a visitor can make multiple requests, then can go to different threads or servers. You would need a way of figuring out if a given visitor has been counted for the current minute or not, and you can't just sum up the per second or per machine number to get a total.

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=5523#respond)

JONATHAN
February 21, 2017 at 8:04 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-5668)

My preferred solution to this involves keeping one bucket per interval (in the example here, that's 60 buckets—1 per minute in an hour). For each hit count, increment the bucket value for the current interval, which will be O(1) for an array of interval buckets.

Separately, clean any bucket that's not the current one frequently enough to ensure that we never get a stale bucket value. Any frequency that is less than the interval of the hit counter should be fine.

Using a linked list seems overly complicated when, assuming that we're not counting for intervals shorter than 1 minute, we're really only storing an array of at most 60 integers.

Example: https://gist.github.com/anonymous/da113f439260daadfbd064e9a45903f6 (https://gist.github.com/anonymous/da113f439260daadfbd064e9a45903f6)

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=5668#respond)

JONATHAN
February 21, 2017 at 8:17 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-5669)

Ah, this isn't a rolling window, though, which I see now is what this question is asking for.

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=5669#respond)

RASHID
February 26, 2017 at 7:50 am (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/#comment-5757)

Do we need to keep count off unique visitors ?
If a user comes to Page1, then Page2, then Page3, do we count as 3 or 1 (first page counts for that time) ?
If unique visitor, shouldn't we keep a hash map of users so the same user doesn't count more than once ?

Reply (http://blog.gainlo.co/index.php/2016/09/12/dropbox-interview-design-hit-counter/?replytocom=5757#respond)

LEAVE A REPLY

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

Comment

Name *

Email *

Website

Post Comment

Design eCommerce Website (Part II) (http://blog.gainlo.co/index.php/2016/08/28/design-ecommerce-website-part-ii/)

Uber Interview Questions – Delimiter Matching (http://blog.gainlo.co/index.php/2016/09/30/uber-interview-question-delimiter-matching/)

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/)