InterviewQuestions

Amazon Confidential

Path: Advertising / D16G / Platform / People / jbiggs /

URL Shortening Service

- Check with interviewee whether they are familiar with URL shortening services such as http://tinyurl.com or http://bit.ly - if not, give example scenario
  - A full list of URL shortening services is available at http://searchengineland.com/analysis-which-url-shortening-service-should-you-use-17204

Requirement Gathering

There are 3 major question topics I look for in the requirement gathering phase, and a few minor ones. The three major ones are:

1. Volume (I'd really like them to ask for both the number of new URL mappings and the number of redirect/lookup requests)
2. Data Retention / Expiration
3. User Authentication

Minor ones are:

1. What sort of data do we want to collect out of this system?
2. What are the latency requirements for requests?
3. What globalization / localization / internationalization should be done for this system?
4. API vs. UI

I don't expect candidates to get all of these - if I score the major ones as 2 points each, and the minor ones as 1 point each (for a total of 10 possible points), I'll generally accept somebody who gets 5/10 or better (that usually works out to 2 of the major 3, and 1 of the minor ones). I'll allow them to come back and ask these questions during the design phase and give credit for asking them at that point, but only if I haven't done any prompting. For example, if I ask the question: "how do you determine the scalability model for your design", they get no credit for then asking about volume (though if they still don't, it's pretty much an automatic "not inclined", even if they otherwise scored high enough).

Design

Ask for high level architecture / system components. I want to hear:

- web server with 2 primary uses:
  - a section for entering in new URL mappings

    - a redirect/lookup service for incoming short URLs
- some sort of data storage

Anything less or further needs some explanation (i.e. I'd want to know why somebody would use a middle-tier for this simple of a system)

Questions

Q: How do you design the data storage for this scale of data and requests?
A: I want to see something along the lines of the following design:

N backend partitions, each with M replicas. Scale N to account for data storage and write volume, scale M to account for number of read requests. All writes should be sent to all M replicas of a given partition, while reads can go to a single replica. Use some function of the short URL to determine which partition to store the mapping in / retrieve the data from. Recognition that the storage on each machine should be something like a hash or b-tree is good - we don't need a relational database here. An answer which uses a BDB file for the storage is a great answer, but I'll accept MySQL, Postgres, etc. If candidate proposes a heavy duty relational database (Oracle, SQL Server, etc.), I'll push them for a low cost solution, but the tendency to go there is already a yellow flag. If the candidate proposes a "NoSQL" solution such as Cassandra, Voldemort, or Riak, I'll give bonus points, but still push them for how to solve the problem from scratch (I want to see that they know how to figure out an algorithm for this, not just that they know of a solution somebody else has designed).

If appropriate given the above solution, I will ask:
Q: Does the short URL need to be a function of the long URL?
A: no - anything but no is a red flag here. Hopefully the candidate will explain why they say no, if not, follow up with a question of why not.

Q: How would you implement the expiration policy (remind them of the policy again, making sure the "minimum" or "at least" portion is stated clearly, but without being obnoxious about it, as it is a hint)?
A: The standard answer is to update a last access timestamp on the record on each lookup request. I don't think anybody has ever realized up front the sort of write rates this would entail. This answer is particularly bad if they used a relational database, due to table locking, but is bad for any solution, given that writes must go to all replicas, and this would absolutely destroy the point of having replicas to distribute the request load among (though there would still be a point to having a replica for redundancy, etc.).

The answer I hope to hear is:

```
if (now > record.last_access_timestamp + X_time)
{
  record.last_access_timestamp = now
}
```

coupled with a batch process which iterates through the records performing an equivalent of:

```
expire_timestamp = now - expiration_policy_time - X_time;
foreach record
{
  if (expire_timetamp > record.last_access_timestamp)
  {
    delete record
  }
}
```

I would be delighted if the candidate were to point out how X_time can then be tuned to balance the write rate and the data storage volume (higher write rate goes with lower data storage volume, lower write rate means you have to persist the data for longer, requiring higher data storage volume), but I don't think I've ever heard anybody state this yet.

Q: What happens when you bring a crashed / out of service replica back online?
A: At a minimum, the candidate at least recognizes that it may have missed a bunch of transactions, and shouldn't be put in rotation until up to sync. The naive answer here is that it should get a full copy of the data from one of the other replicas in its partition, but that has its own issues (by the time you've got all that data you're out of sync with what's happened since then).

The answer I hope to hear is:
Each replica should be keeping a record of monotonically increasing transaction ids for each write/delete to enable replay capabilities. When a replica comes back up, it should immediately begin queuing incoming transactions (but not necessarily applying them), then look at its most recent (highest) applied transaction id, and incrementally request chunks of transactions from other replicas in its partition greater than this transaction id and less than the first received queued transaction, and apply them, until no transactions remain in that range. The replica should then begin applying transactions from its queue, and once caught up to within X time behind real time (where X is defined as an availability SLA), should put itself back into the read rotation.

Bonus points if they clean up after themselves with a process which periodically scans for maximum applied transaction id across all M replicas (whether currently up or not, if not all are up, then no clean up possible), and delete transaction records less than the maximum found. If a replica is down for too long, it should be removed from consideration as a replica for the partition, so that clean up can proceed, and later added in as a new replica host for the partition.

New replica hosts for the partition should be introduced by taking another replica out of service (both reads and writes), allow any queued transactions to apply, and then fully copy its data (including transaction records) to the new replica host. Once the copy has been completed, both replicas should be reintroduced in the manner described above for out of service replicas.

I judge candidates based on their ability to come up with reasonable designs (not necessarily exact replicas of my desired solution), and on how many of the questions they get through. Most candidates do not make it past the expiration policy question before phone screen / interview time runs out. The few that have made it to the "out of service replica" problem have not produced answers better than the minimum problem recognition and naive solution.

Lift Analysis

This is typically an in-person interview question.

Problem intro

In Display Advertising, as in most advertising related businesses, we want to be able to show the advertiser that the ad dollars they are spending are generating results for them. The metric that we use for this is what we call "lift" - the ratio of the purchase rate of a set of users who saw the ad to the purchase rate of a similar set of users who are differentiated only in that they did not see the ad. In the industry, this is often performed via mechanisms such as Public Service Announcement ads shown to a random set of users who were chosen to see the standard ad. Unfortunately, this is costly and rather wasteful (explain why, if necessary).

Because we know every user who saw each ad (and who didn't), we could instead identify a set of users who met the criteria to see the ad, but who did not actually see the ad. We could then compare the purchase rate of users who saw the ad to that of users who did not see the ad.

Q1: Do you see any flaws in this methodology?

Typically, candidates don't. So, I give them a flaw (see below)... If they do find a flaw (and they can explain it), they're already well ahead of the curve, and I ask them how they could account for the flaw/flaws that they pointed out, while maintaining the same high level methodology (i.e. synthesizing a "similar" control group out of the set of all users who did not see the ad)

User Activity Level Chart

At this point, I typically draw a chart of "probability of seeing ad" on the Y axis vs. "# of page views on Amazon" on the X axis. This chart starts at the origin (0,0), and grows to asymptotically approach 100% (or some cap, if there is a % of users held out) as # of page views grows. I then point out how users who see the ad will tend to skew towards high #'s of page views on Amazon compared to the average Amazon user, and users who have not seen the ad will either skew towards lower #'s of page views on Amazon then the average user (or if picking only out of a hold out, the users who have not seen the ad will look just like average users, which is still a problem, since those that were exposed to the ad are not representative of average). I then furthermore point out that (in general) users with higher levels of activity on Amazon are more likely to purchase any product on Amazon than users with lower levels of activity on Amazon. There's an opportunity to ask a question here (Can you explain why this matters?), but I usually skip past that, and talk about how this means that users who have seen the ad will tend to skew towards higher purchase rates regardless of having seen the ad, than users who did not see the ad, and that therefore the strict ratio of purchase rates of exposed users vs. qualified non-exposed users will tend to over-report lift.

Q2: How do you account for the activity level bias?

This is really the meat of the question - I'd love to see all candidates actually do some math here, but find that few actually do... What I'm hoping for is that the candidate will at least verbalize some amount of "I want to adjust or normalize the control set to have a similar page view composition as the treatment set" (recognizing full well I'll never get somebody using those terms...). Unfortunately, this tends to be a make or break question, and I find many candidates need some hinting. For a first hint, I just give a gentle reminder that the problem we have is that there is too much of an effect from the low page view control users relative to the exposed/treatment set compared to the high page view control users relative to the exposed/treatment set. A second hint is to draw a vertical slice in the chart representing users who have a particular # of page views, and point out that at that particular # of page views, we could (within the framework of the question, at least) safely calculate a lift ratio, as both the exposed and control users have the exact same profile in terms of # of page views on Amazon. I might extend this out for the candidate to point out that they could come up with a ratio across every page view count we recorded. As one final hint, I talk about how they can't just average these lift numbers calculated per # of page views slice, since there's presumably very different numbers of users who saw the ad within each slice, etc. At this point, if they don't get the answer (taking a weighted average of these lift numbers by # of users who saw the ad in each slice), I give up on the question.

Q3: How do you extend this across potentially hundreds of guaranteed campaigns?

note: I really don't expect _anyone_ to get this (though I have high expectations for ML and PE candidates on this part) - this question is more about selling the candidate that has succeeded at Q2 on the complexity of problems we're thinking about and solving here...

I start this question by talking about how, while the ad we're trying to measure might be targeting users who are in-market for digital cameras, a completely unrelated campaign might be targeting gourmet coffee lovers. I point out how, in an extreme case, with guaranteed ad serving, we might end up never serving any other ads than the gourmet coffee lover targeted ad to users who fall into the gourmet coffee lover segment. This would then mean that the set of users who saw our digital camera ad would not include any gourmet coffee lovers, though our control group does. If gourmet coffee lovers who were also in-market for digital cameras were any more or less likely to buy digital cameras than the average user who was in-market for digital cameras, this creates a skew between our control and treatment sets. In one scenario, if gourmet coffee lovers who were in-market for digital cameras are significantly more likely to buy a digital camera than the average in-market for digital camera user, this could cause it to appear that showing the ad actually decreased the likelihood that a user would then purchase the advertised digital camera.

I then usually skip the obvious next question (how do you now account for both the # of page views skew as well as the gourmet coffee lover skew), and just point out how with hundreds of concurrently running campaigns, each with its own targeting and varying degrees of causing users to not see the digital camera ad, this problem becomes rather impossible to deal with via further bucketing and applying a weighted average.

As I said, I don't really expect anyone to get through this, but it's always interesting to see the thought processes and approaches taken by candidates who are willing to try to tackle this problem. My personal preference is to use a similarity measure with iterative permutation of the control set to generate a synthetic control set that is as similar to the exposure/treatment group as possible (look for my write up titled "Lift Analysis via Cosine Similarity (https://w.amazon.com/index.php/Advertising/D16G/Platform/People/jbiggs /LiftAnalysisCosineSimilarity) " on the wiki to read more about this proposed solution). As far as I recall, only one candidate has ever proposed an answer along these lines.