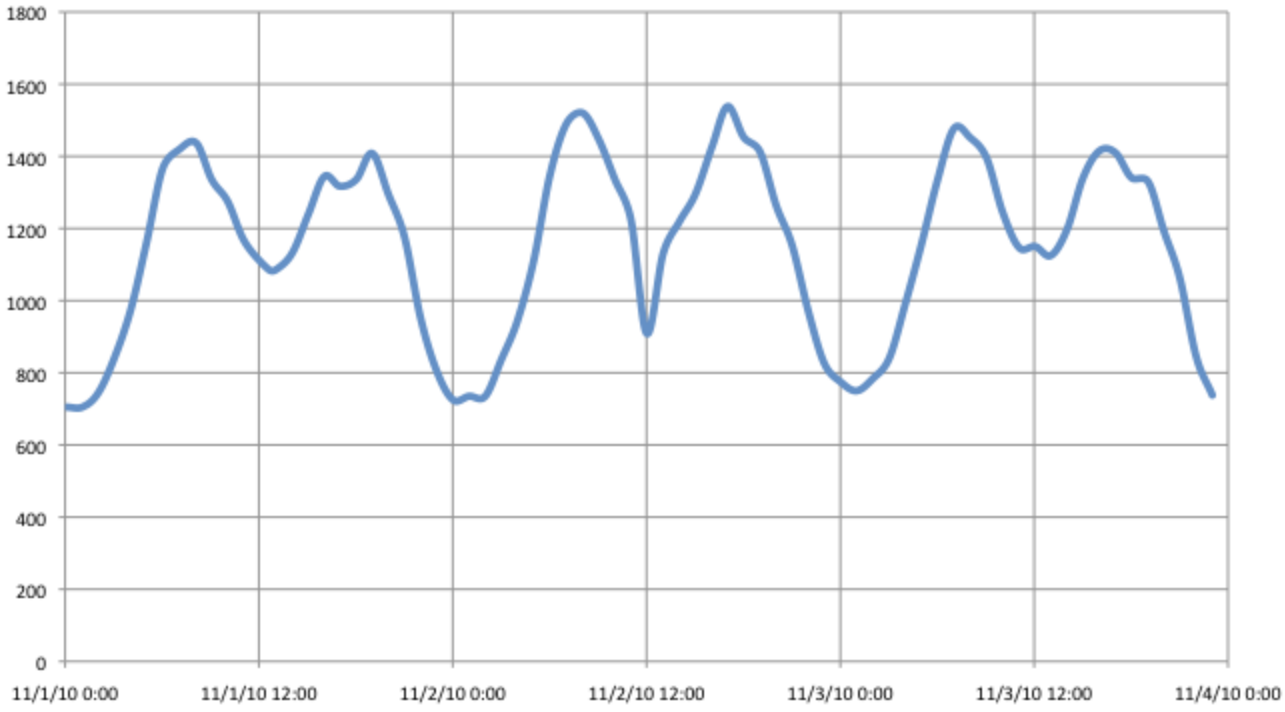
Tracking Twitter's Growth after Snowflake

Jim Bumgardner, DigiSynd

From March 2008 thru early November 2010, it was possible to track the total number of tweets being added to Twitter by looking at the id numbers of new tweets. With a few exceptions (such as days when Twitter restarted the server and skipped large chunks of id numbers), Twitter assigned id numbers sequentially. For most short time intervals, you could measure the rate that new tweets were created by comparing the time stamps and id numbers of successive tweets, using the following formula:

$$t = \frac{later\_id - older\_id}{later\_seconds - older\_seconds}$$

Here is a graph showing overall Twitter traffic (measured in tweets per second) for the first few days in November, using this method hourly on a 100 tweet sample.



On November 4th 2010, shortly after 2pm PDT, Twitter switched to a new ID system called Snowflake, citing scalability issues. Snowflake was first announced in June:

http://engineering.twitter.com/2010/06/announcing-snowflake.html

and was further discussed here:

http://groups.google.com/group/twitter-development-talk/browse_thread/thread/6a16efa375532182

In a follow up message in the above thread, Twitter developer Matt Harris describes the makeup of a snowflake ID as follows:

> "A Snowflake ID is composed:
> * 41bits for millisecond precision time (69 years)
> * 10bits for a configured machine identity (1024 machines)
> * 12bits for a sequence number (4096 per machine) "

Sample Python code to extract these elements given an id:

```
seq = id & 0x0FFF
machine = (id >> 12) & 0x3F
timestamp = (id >> 22)
```

An analysis of the ids shows that to date, Twitter has been using three "machines" to process snowflake ids, and since 11/11, the load appears to evenly spread across all the machines in use.

When a snowflake machine assigns a new id, it gives the id a sequence number which corresponds to how many ids have been created by that machine within the same millisecond. For any specific millisecond timestamp, and any specific machine, the first id created has a sequence number of 0, the second id gets a sequence number of 1, and so on.

The key insight is that non-zero sequence numbers have similar properties to hash table collisions (or the well known "birthday problem"). Since there are three machines, each unique second has 3*1000=3000 "slots", or unique hash values. When the rate at which id numbers is being created is low, the number of collisions (or non-zero sequence numbers) is correspondingly low. As the rate of id issuance increases (and the number of machines creating those numbers remains the same), the number of collisions should increase. Hence, given a large enough sample, we can use the ratio of collisions to non-collisions to estimate overall Twitter traffic.

Hash table collisions are a well understood problem. The equation that predicts how many hash table collisions will occur for a given number of slots (d), and a given number of tries (n) is

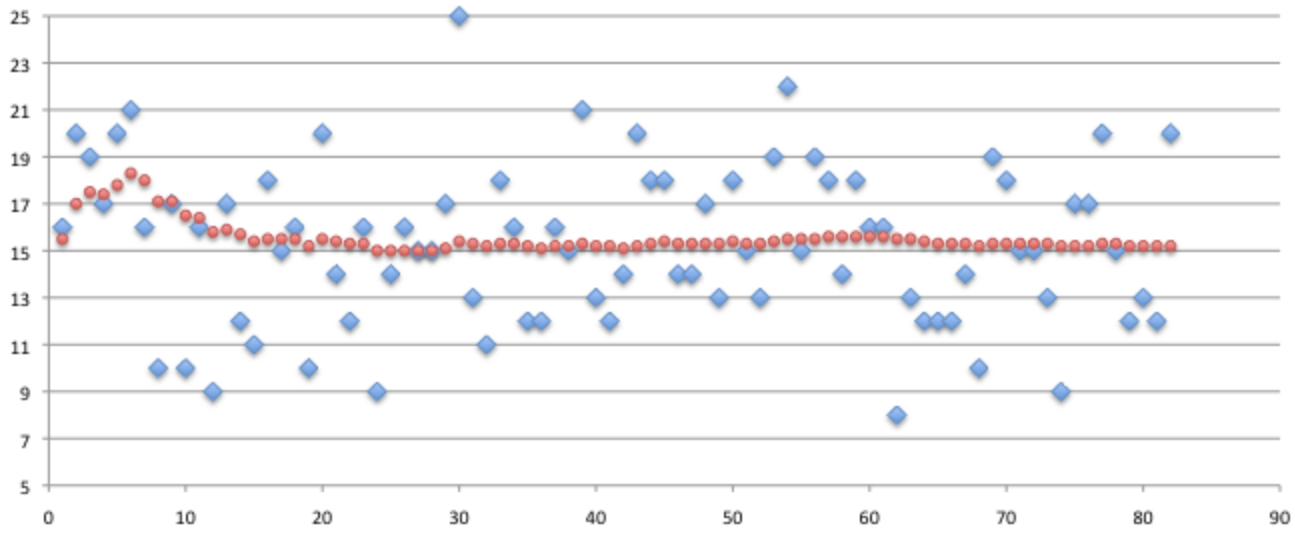
$$C = n - d + d\left(\frac{d-1}{d}\right)^n$$

http://en.wikipedia.org/wiki/Birthday_problem#Collision_counting

Note that the first equation for computing collisions does *not* provide the number of duplicate hash values. Rather it predicts, for a given number of tries (n) how many times a number will duplicate a number already chosen. This value is smaller than the total number of duplicate hash values, since for each set of identical values, it doesn't count the first generated one in the set.

In the case of Twitter Snowflake id's machines, assuming an evenly distributed rate of tweets (a big assumption, see below) C is the number of non-zero sequence numbers issued per second, n is the number of tweets issued per second, d is the number of available slots per second, or 1000 X machines. Currently, with three machines d = 3000, however, for any set of tweets, it is easy to count the number of machines in use.

I estimate that a 3000 tweet sample is sufficient to get to within plus or minus 1 from the average collision rate, although even larger samples are preferable. Here is a typical Monte Carlo run showing a series of 100 tweet samples (the maximum number of tweets returned by a single call to Twitter's search API), and the running average number of collisions given a rate of 1000 tweets per second (predicted rate should be 14.9).



(Fig 2. Blue dots represent the number of collisions in each run of 100 tweets. Red dots represent the running average number of collisions.)

Here's python code for a simulation showing how well the C equation predicts collisions at different tweet rates. Feel free to play with it... It assumes an even distribution of tweets over the full second. Note that current tweet rates are on the order of 100 million per day, or roughly 1100 per second.

http://pastie.org/private/rtt7bjyejiwbywhmjtwb0q

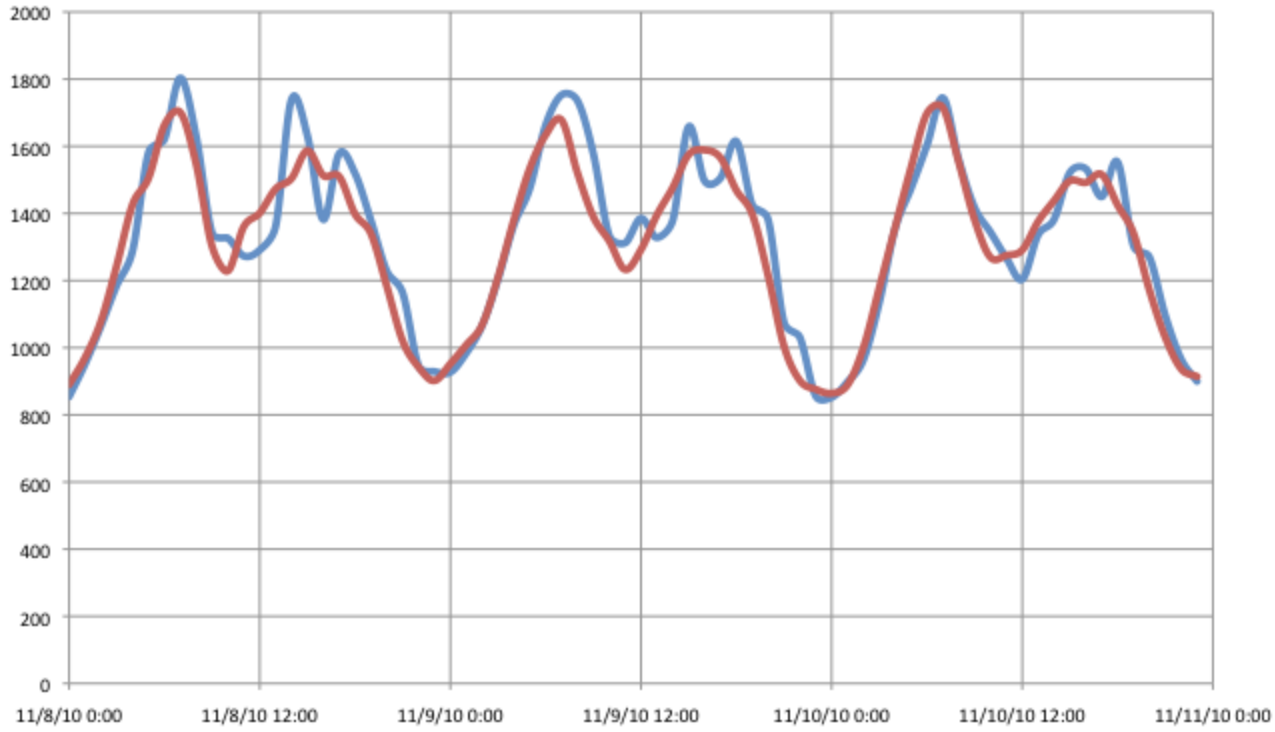
To estimate the overall rate of Tweets, I pull a set of 3000 or more recent tweets. I count the number of non-zero sequence numbers. I also count the number of machines in use. I now have values for C and d in the above formula. I can then use the above equation to derive a value for N, which is the current rate of tweets.

It is easy enough to use the successive approximations of the above equation to find a suitable value for N (given D and C). David Horn has also provided the following solution for n:

$$n = \frac{LambertW(d*ln(\frac{d-1}{d})*(\frac{d-1}{d})^{C+d})}{ln(\frac{d-1}{d})} + C + d$$

Where LambertW is the Lambert W function:
http://docs.scipy.org/doc/scipy/reference/generated/scipy.special.lambertw.html

Here is a sample graph using the post-snowflake method showing the estimated Tweets per second for November 8th-10th using approx 3,000 tweets per hourly measurement (blue line) and 30,000 tweets per hourly measurement (red line).



Known Issues

This method assumes that during any given second, the rate at which tweets are created is

constant. In reality, it is quite possible that tweets could to be created in short bursts of activity, followed by gaps of inactivity. These bursts might cause the collision rate (and hence the computed tweet rate) to be artificially inflated. I believe most natural deviations from a statistically flat distribution will cause the collision rate to rise.

I believe we are seeing as much as 20% inflation caused by this problem, hence the significantly higher apparent tweet rate in the 11/8 graph, although we can still accurately measure twitter's growth (or decline).

Jim Bumgardner 11/15/2010