UBlog benchmark

Ricardo Vilaça, Rui Oliveira and José Pereira
Universidade do Minho
{rmvilaca,rco,jop}@di.uminho.pt

November 29, 2010

1 Test workload

UBlog benchmark mimics the usage of the Twitter social network.

Twitter is an online social network application offering a simple micro-blogging service consisting of small user posts, the *tweets*. A user gets access to other user tweets by explicitly stating a *follow* relationship.

The central feature of Twitter is the user *timeline*. A user's timeline is the stream of tweets from the users she *follows* and from her own. Tweets are free form strings up to 140 characters. Tweets may contain two kinds of tags, user mentions formed by a user's id preceded by @ (e.g.. @john) and hashtags, arbitrary words preceded by # (e.g.. #topic) meant to be the target of searches for related tweets.

Our workload definition has been shaped by the results of recent studies on Twitter [3, 4, 2]. In particular, we consider just the subset of the seven most used operations from the Twitter API [5] (Search and REST API as of March 2010):

$Tweet^*$ `statuses_user_timeline(String userID, int s, int c)` retrieves from userID's tweets, in reverse chronological order, up to c tweets starting from s (read only operation).

$Tweet^*$ `statuses_friends_timeline(String userID, int s, int c)` retrieves from userID's timeline, in reverse chronological order, up to c tweets starting from s. This operation allows to obtain the a user's timeline incrementally (read only operation).

*Tweet** `statuses_mentions(String userID)` retrieves the most recent tweets mentioning userID's in reverse chronological order (read only operation).

*Tweet** `search_contains_hashtag(String topic)` searches the system for tweets containing topic as hashtag (read only operation).

`statuses_update(Tweet tweet)` appends a new tweet to the system (update operation).

`friendships_create(String userID, String toStartUserID)` allows userID to follow toStartUserID (update operation).

`friendships_destroy(String userID, String toStopUserID)` allows userID to unfollow toStopUserID (update operation).

For the implementation of the test workload we consider a simple data model of three collections: `users`, `tweets` and `timelines`. The `users` collection is keyed by userid and for each user it stores profile data (name, password, and date of creation), the list of the user's followers, a list of users the user follows, and the user's tweetid, an increasing sequence number. The `tweets` collection is keyed by a compound of userid and tweetid. It stores the tweets' text and date, and associated user and topic tags if present. The `timelines` collection stores the timeline for each user. It is keyed by userid and each entry contains a list of pairs (tweetid, date) in reverse chronological order.

In a nutshell, the operations listed above manipulate these data structures as follows. The `statuses_update` operation reads and updates the user's current tweet sequence number from `users`, appends the new tweet to `tweets` and updates the timeline for the user and each of the user's follower in `timelines`. The `friendships_create` and `friendships_destroy` operations update the involved users records in `users` and recomputes the follower's `timelines` adding or removing the most recent tweets from the followed, or unfollowed, user. Regarding the read only operations, `statuses_friends_timeline` simply accesses the specified user timeline record in `timelines`, `statuses_user_timeline` accesses a range of the user's tweets, and `statuses_mentions` and `search_contains_hashtag` the `tweets` collection in general.

The application is firstly initialized with a set of users (that remains unchanged throughout the experiments), a graph of follow relationships and a set of tweets.

Twitter's network belongs to a class of scale-free networks and exhibit a small world phenomenon [3]. As such, the set of users and their follow

Table 1: Probability of Operations

Operation	Probability
search_contains_hashtag	15%
statuses_mentions	25%
statuses_user_timeline	5%
statuses_friends_timeline	45%
statuses_update	5%
friendships_create	2.5%
friendships_destroy	2.5%

relationships are determined by a directed graph created with the help of a scale-free graph generator [1].

In order to fulfill `statuses_user_timeline`, `statuses_friends_timeline` and `statuses_mentions` requests right from the start of the experiments, the application is populated with initial tweets. The generation of tweets, both for the initialization phase and for the workload, follows a couple of observations over Twitter traces [4, 2]. First, the number of tweets per user is proportional to the user's followers [4]. From all tweets, 36% mention some user and 5% refer to a topic [2]. Mentions in tweets are created by randomly choosing a user from the set of friends. Topics are chosen using a power-law distribution [3].

Each run of the workload consists of a specified number of operations. The next operation is randomly chosen taking into account the probabilities of occurrence. The defaults values are depicted in Table 1. To our knowledge, no statistics about the particular occurrences of each of the Twitter operations are publicly available. The figures of Table 1 are biased towards a read intensive workload and based on discussions that took place during Twitter's Chirp conference (the Twitter official developers conference, e.g.. `http://pt.justin.tv/twitterchirp/b/262219316`).

The defined workload may be used with both key-value stores and relational databases. Currently, there are implementations for Cassandra, Voldemort, and MySQL.

References

[1] A.-L. Barabási and E. Bonabeau. Scale-free networks. *Scientific American*, 288:60–69, 2003.

[2] Danah Boyd, Scott Golder, and Gilad Lotan. Tweet tweet retweet: Conversational aspects of retweeting on twitter. In IEEE Computer Society, editor, *Proceedings of HICSS-43*, January 2010.

[3] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65, New York, NY, USA, 2007. ACM.

[4] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 19–24, New York, NY, USA, 2008. ACM.

[5] Twitter. Twitter api documentation. http://apiwiki.twitter.com/Twitter-API-Documentation, 2010.