HW2: Twitter performance with Redis

DS 4300: Large-Scale Storage and Retrieval / Prof. Rachlin

DESCRIPTION

In this assignment we'll follow the lead of the Twitter engineers who recognized the scaling limits of a relational database by re-implementing our Twitter API to use Redis, a Key-Value store.   You'll have some design choices to make in terms of what distinct keys you manage, and whether to put raw data into each value, or simply references to other keys in order to reduce storage overhead. As before, we are interested in the performance of two basic operations (measured in API calls per second):

A) **postTweet:** Post a tweet by a random user

B) **getTimeline:** Obtain the home timeline of a random user (i.e., for a random user, get the 10 latest tweets of all the users that your randomly chosen user follows.)

IMPLEMENTATION

If you structured your homework 1 correctly, you should not have to change very much in your main driver program which performs the timing analysis. You will be calling the same Twitter API methods, but now they are backed by a Redis implementation.

ANALYSIS

There are two strategies for implementing the postTweet and getTimeline operations that you might explore. The first strategy is optional. Do this if you have time. The second strategy is what the Twitter engineers ultimately decided to do and is required for the homework.

Strategy 1 (OPTIONAL): When you post a tweet, it is a simple set operation, where the key is the tweet ID (perhaps "Tweet:12345") and the value is the contents of the tweet.   The getTimeline operation will require that you look up the tweets of each user being followed and that you construct the home timeline on the fly.   (Posting tweets should be very fast while home timeline requests take longer.)

Strategy 2 (REQUIRED): As you post each tweet, you copy the tweet (or a reference to the tweet) to the user's home timeline automatically. Your write performance should now be slower, but since the timeline is ready and waiting, fetching the timeline should be a much faster operation.   Let's find out how much faster!

As before, report the postTweet and getTimeline performance (API calls per second). Write up a summary report (1 page) and record your final results on the Google Sheets document.

GRADING

Submit your code, including a documented Twitter API and its Redis implementation.

You will be graded on the quality and completeness of your code and the design and implementation of your Redis-based Twitter API.