DS 4300 Homework2

By Xiang Zhu and Boyuan Li

Results

Twitter Performance with Redis:

	Strategy 1	Strategy 2
postTweet	5125.8 tweets/sec	941.8 tweets/sec
getTimeline	2.45 timelines/sec	26.73 timelines/sec

Twitter Performance with MySQL:

	Rates
postTweet	971.42 tweets/sec
getTimeline	0.37 timelines/sec

Abstract

In this report, we compare the performance in operations of twitter, like posting tweets and randomly retrieving a home timeline, when using a relational and non-relational database. We use Python to mimic an interface to interact with Redis database. Comparing this implementation and performance to what we did by using MySQL in the first assignment, we find that using Redis, a non-relational database is way more efficient than MySQL, a relational database.

Introduction

We found that using MySQL to simulate operations of Twitter is time-consuming. Besides the low performance of hardware such as CPU, we believed that it was also due to the relational database. Therefore, we use Redis as our database to test the performance in operations of Twitter.

Method

We worked on Mac to use Redis, and we used abstract class to achieve the implementation of Twitter API since there is no interface in Python. We created a class called TwitterAPI within some methods such as postTweets, addFollowers, getTimeline and so on. Then we created another class called RedisAPI which inherits from TwitterAPI. Based on what we did in

assignment 1, we already have two csv files storing followers/followees and tweets, so we created three methods (functions) , add_followers, add_followees, post_tweets , to import data from csv files to Redis database. For follower/followee relationship, we used "followers: user_id" as a key and a set of use ids followed by this person as values. Moreover, "followees: user_id" is the key with a set of user ids following this user.
As for postTweets and getTimeline, there are two strategies to do that. We use a boolean strategy (strategy 1 = False, strategy 2 = True) to implement these functions. For strategy 1, every tweet is posted as an addition of a key and a value. This way is easy and faster, but as for getTimeline, firstly every followers of the user will be found, then all tweets those followers posted are constructed as the timeline, which is time-consuming. For strategy 2, when a tweet is posted, it is directly saved to the timelines of those people who follow the user, therefore, postTweets costs more time, but getTimeli is quick since the timeline is already created automatically and waiting.

Analysis
Based on the performance result above, for strategy 1, postTweets is definitely fast since the way it is stored is key-value-store. Compared with the rate of posting tweets by using MySQL, using Redis is around 50 times faster. And the rate of getTimeline in Redis is still faster than that in MySQL,though the speed is slow. As for strategy 2, postTweets rates are nearly the same, and Redis's getTimeline rate is way faster than MySQL's rate.

Conclusions
These two strategies have their own advantages. Strategy 1 is a good choice for posting tweets. And if you focus on getting timelines, then strategy 2 supports it a lot. Therefore, we think it depends on which operation people want to focus on. As we discussed in class, Twitter has 6000 to 10000 tweets posted per sec, strategy 1 is the best choice. However, Twitter has to deal with 200,000 to 300,000 timelines per sec. Strategy 1 is not a good idea. We think we should take the capability of computers into consideration. The computing power of a personal computer is low honestly. Twitter should have a stronger server computer, so the need of posting tweets should be easily met. Then I recommend using strategy 2 since the rate of getting timelines is 10 times faster based on the performance shown on our laptop.

Author Contributions
For assignment 1, we worked individually. For assignment 2, We are working in a group of two in order to learn more from each other. Both of us did the coding part, and we separated the work of paper. One did results, abstract, and introduction. The other one did method, analysis and conclusions.