DS 4300: Assignment 1
Twitter: The Early Years

DESCRIPTION

When Twitter first started out, its engineers used MySQL as a backend database. As the service grew in popularity, they were forced to abandon MySQL in favor of a NoSQL key-value store (Redis). In the assignment and the next, you will play the role of a twitter engineer trying to understand the performance limitations of MySQL and the potential benefits of a key-value store.

There are two key functions of twitter. First, it allows users to post tweets. Second, it allows a user to see all the tweets posted by people that user is following. This set of tweets – which the user sees when he/she opens up the twitter app on a smart phone – is known as the user's home timeline.

In this assignment you'll push MySQL to its limit by seeing how fast you can post tweets and retrieve home timelines.

DATABASE AND DATA SETUP

1. Implement a simple relational database to manage users and their tweets. Your database should have two tables:

TWEETS – The tweets posted by users

tweet_id	long
user_id	long
tweet_ts	datetime
tweet_text	varchar(140)

FOLLOWERS – Who follows whom

user_id	long
follows_id	long

You don't need to worry about a user table or a hashtag table.

2. Write a tweet generator that creates random tweets and saves them to a file. The tweets can be random words from some corpus of your own choosing, or they can be complete gibberish. But your text file of tweets should contain at least 1 million tweets.

3. Generate random user/follows records for the FOLLOWERS table. How many different users should we have for a million random tweets? How many followers should each user have? Document your assumptions!

PERFORMANCE TESTING

1. Write one program that reads pre-generated tweets from the file and loads them into MySQL. Keep track of how long it takes to load all of the tweets. This program simulates users posting tweets. How many tweets can be posted per second? (Twitter receives 6-10 thousand new tweets per second. Can MySQL keep up?)

2. Write a second program that repeatedly picks a random user and returns that user's home timeline. For example, if user A follows X, Y, and Z, then A's home timeline consists of the 10 most recent tweets posted by X, Y, or Z. This process simulates users opening the twitter app on their smartphone and refreshing the home timeline to see new posts. How many home timelines can be retrieved per second? (Twitter users worldwide collectively refresh their home timeline 200-300 thousand times per second. Can your program keep up?)

ANALYSIS AND REPORTING (Be Creative – but here are some general guidelines)

1. In a table, document your hardware configuration (CPU speed, number of cores, RAM, Disk etc.) and data model assumptions: number of users, number of tweets per user, distribution of the number of followers per user, or any other parameters or database settings that might affect your results.

2. Discuss the implications of your system parameters on performance. For example, how would a random-vs-skewed distribution of followers impact your results? Does your performance change as your database fills up with records / values? If I had asked you to generate 1 billion tweets, would that have been a bad idea? Is so, why?

3. Calculate the theoretical optimal performance. In other words, compare inserting records in a database to simply writing the records to disk. And compare doing a query against a relational database to retrieve the home timeline to simply reading 10 tweets from disk. How close to optimal performance are you?

SCORING

You will be graded on your MySQL implementation, the quality of your code, your experimental design, and the thoroughness of your analysis. *We will compare results in class and I may ask several of you to present your analysis to the rest of the class. Don't worry if your performance is really slow – or slower than your fellow students. This isn't a competition and there are many factors that can influence the results. Let's figure out what those factors might be!*

IMPORTANT: To facilitate in-class discussion, please clearly report in a table:

- *Tweets inserted per second (average)*
- *Home timelines retrieved per second (average)*