Julia Geller, Ceara Zhang

Professor Rachlin

DS 4300

22 January 2024

**Analysis and Reporting for Twitter Relational Database**

**Hardware**

* Computer: Macbook Pro
* Processor: Apple M1 Max chip
* RAM: 32 GB
* Cores: 10

**Software:**

* MacOS: Sonoma 14.2.1
* Database: MySQLWorkbench 8.0.28
* Language: Python 3.10.9
* Libraries: List, random, mysql.connector, pandas, os, time, datetime

**Results**

|  |  |
| --- | --- |
| **API Method** | **API Calls/Sec** |
| post\_tweet | 3800 |
| get\_home\_timeline | 45 |

**Analysis**

The results I obtained were within the same magnitude of most other results produced by my classmates with similar configurations and was on par with class expectations for post\_tweets. The M1 Max chip is a powerful processor, contributing to the high performance of the system. Having more RAM also allows for efficient handling of data and concurrent operations. For getting home timelines, we initially wrote a query with an explicit join statement, but this significantly increased the retrieval time as these types of queries are expensive. In our final version, we simulate a join operation by first obtaining the ‘follows\_id” for a given user and then retrieving tweets for those users, which increased run time significantly.

The code also implements database indexing. Since we are dealing with a high volume of tweets, this ensures that the ‘tweet’ and ‘follows’ tables are appropriately indexed to optimize query performance. In our analysis, the indexing made a significant improvement for getting home timelines, increasing the average from 5 per second to 45 per second.

For further improvements, prepared statements could be investigated, as they provide several benefits including preventing SQL injection, improving performance, security, and code readability.