Julia Geller, Ceara Zhang

Professor Rachlin

DS 4300

3 February 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 For Strategy 1** | **Calls/Sec for Strategy 2** |
| post\_tweet | 15,500 | 8,800 |
| get\_home\_timeline | 85 | 91 |

**Analysis**

In the Twitter Redis Database simulation, while posting tweets using both Strategy 1 and Strategy 2 demonstrates satisfactory performance, the home timeline retrieval lags behind expectations significantly, recording approximately 90 timelines per second instead of the anticipated 1000s. Possible reasons for this discrepancy include suboptimal data modeling, storage, and Redis operations. Strategy 1 constructs the home timeline on-the-fly, potentially causing computational overhead, while Strategy 2, despite pre-populating timelines, may still have room for optimization in the retrieval process.

To address these issues, we could consider implementing batch retrieval with Redis pipelines to reduce round-trip operations, optimizing Redis operations, exploring alternative data serialization formats (e.g., JSON), introducing concurrency in home timeline retrieval, optimizing database indexes for query patterns, and implementing a caching mechanism to store frequently accessed home timelines.