DS 4300 HW1 Analysis

Ethan Gilworth and Rithesh Kayathi

Report on Database Performance

Using SQL to manage a large scale application like Twitter could be inefficient for a number of reasons:

Scalability: As the number of users and tweets on Twitter increases, the amount of data that needs to be stored and queried also increases. SQL databases are not as well-suited to handling large amounts of data as other technologies, such as NoSQL databases or distributed systems.

Real-time data: Social media platforms require real-time data processing and analysis which SQL databases may not be able to handle as efficiently as other technologies like stream processing, event-driven architectures, etc.

Flexibility: SQL databases are often less flexible than other technologies when it comes to handling unstructured or semi-structured data, and may not be able to keep pace with the rapidly evolving requirements of a large scale application like Twitter.

All these reasons make it clear that even though SQL databases have been a trusted and popular way of managing data for a long time, they may not be the best fit for a large scale application like Twitter.

Here are some things that might have limited our results.

1. Lack of batching for database inserts: Inserting records one at a time can be much slower than batching multiple inserts together and committing them to the database in a single transaction
2. Lack of secondary indexing: Secondary indexes can greatly improve the performance of database queries, especially when retrieving large amounts of data. We did not use them, as I'm not sure what they are, to be honest.
3. Use of SQL Joins: Joining multiple tables in a query can be an inefficient operation, especially when the tables being joined are large. Perhaps approaching it from a different angle would be better, related to the getFollowees and getTweets method described in the homework.
4. Hardware limitations: My MacBook Pro's 8 GB of RAM and Apple chip are top of the line in terms of computations per watt, but in raw computation power per second there are better chips out there.

5. MySQL settings: The default MySQL settings may not be optimized for the type of workload you are trying to run on my MacBook Pro, and tweaking these settings may have improved performance. It seems as though those who used InnoDB generally performed better than those who used default settings.

6. Database Scale: This was a fairly large database, so I think that the joins we used took quite a while to stitch together.
7. Network Latency: While our internet is fast, it is not fully optimal.
8. Disk I/O: The performance of the relational database on my macbook pro might be limited by the speed at which data can be read from and written to the disk.
9. Data Modeling: The data model you used for the twitter mock version might not be optimal for your use case leading to performance issues