Group: VanGar
Noah Van Hooser
Jennifer Garza

CS 327E: Final Project

Section 1: Overview

A relational database management system (RDBMS) serves to create, update, and administer relational databases. The purpose of this project was to test the performance of a RDBMS under average caseload and with load spikes. In order to create the databases that would be tested, our group used MySQL Server 8 as the data definition language, DDL. We utilized MySQL Connector/Python in order to employ Python commands that would allow for the development, testing, and management of the database application.

The records were obtained and extracted from two files. The file 'us-500.csv' contained 500 rows and served as the initial sample data. The file 'us-1000000.csv' contained 1 million records that would then be loaded into the database and used to assess the RDBMS performance.The project is divided into two sections: a baseline run and a batch run. We measured the latency of writing 8 million records onto the database before and after making software and hardware optimizations.

Section 2: Baseline Run

For the baseline run, we had to extract the data files onto the database we created through a sql script. During the baseline run, we utilized MySQL connector's `execute()` method to insert each row. We then used the `%%timeit` function in order to obtain the total execution time of our code. One of the technical challenges we encountered was the fact that we could not produce an output of the full 8 million records due to the amount of time it would take to run the code. We timed how long it took to insert 250,000 records into the database using a timer, and it took an average of 2 minutes and 43 seconds per every 5,000 records. At that rate, the amount of time it would take to load 8 million records onto the database would be 9 hours 49 minutes.

Section 3: Batch Runs

In the second half of our project, downloaded the `max_allowed_packet` variable. Rather than using the `execute()` method to perform one insert statement at a time, we utilized the `executemany()` method to create batch inserts to the database in increments of 5000, 50,000, and 500,000 which we noticed was a significant improvement from our baseline run. Our batch run results were as follows:

Small batch size (5,000):

```
980000 records inserted successfully into Person table
985000 records inserted successfully into Person table
990000 records inserted successfully into Person table
995000 records inserted successfully into Person table
1000000 records inserted successfully into Person table
MySQL connection is closed
51.1 s ± 162 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Medium batch size (50,000):

```
850000 records inserted successfully into Person table
900000 records inserted successfully into Person table
950000 records inserted successfully into Person table
1000000 records inserted successfully into Person table
MySQL connection is closed
42.6 s ± 393 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Large batch size (500,000):

```
MySQL connection is closed
<mysql.connector.connection_cext.CMySQLConnection object at 0x7fcde51a1ed0>
500000 records inserted successfully into Person table
1000000 records inserted successfully into Person table
MySQL connection is closed
40.6 s ± 413 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Section 4: Hardware Upgrade Runs

We then performed hardware upgrades. This entailed changing the machine series from N1 to C2 as well as the machine type from n1-standard-8 to c2-standard-30. In essence, this upgraded our machine, providing 30 CPUs and 120GB of remote access memory (RAM). After performing the hardware upgrade, we re-tested with the large batch size, and obtained an average of 36.1 seconds ± 443 ms per loop. Lastly, we ran a final test with the high-bandwith server and obtained an average of 38.6 seconds ± 538 ms per loop.

Section 5: Potential Future Improvements

We would have liked to investigate the potential of running a paired t-test to compare the latency and performance of a MySQL database to that of a PostgreSQL database before and after both were given hardware changes. A paired t-test would build on this study by allowing us to compare both groups side by side and analyze if and how relational databases differ from one another, and if the difference is significant.