CS 223

PROJECT 1

REPORT

Utkarsha Ganla - 61447048
Tushar Kulkarni - 20648620
Arundhati Sawant - 29525453

1. Design of the simulator

a. Preprocessing the information

Input to the system are three files observations, semantic observation and queries. Each file consists of operations. Operations are not sorted according to timestamp. In the pre-processing phase, the following steps are performed:

1) Parse the file.
2) Extract the timestamp using regular expressions.
3) Map the operations occurring in one timestamp to the timestamp value and store the list of operations in a TreeMap.
4) TreeMap maintains the key value pair sorted by keys.
5) Use serialization to write the entire TreeMap object to a file.

b. Simulation

The given set of queries spans over 20 days. To simulate the actual execution, 20 days have been mapped to 20 minutes. Therefore, all operations happening in 3 minutes are mapped to 125 milliseconds. This can be fine-tuned to milliseconds. Every second a scheduler inserts the operations happening at that time instant (mapped time instant) into a queue.

c. Thread Creation

A thread pool is implemented to avoid overhead in creating new threads every time. The number of threads in the thread pool is the MPL. An available thread polls the queue to check if transactions are available. Once a transaction (group of operations) becomes available, the thread begins executing the transaction at the given isolation level.

d. Transaction definition

For Read Committed, Read Uncommitted, Repeatable Read, the time duration of 3 minutes is mapped to 1 second. For Read Serializable, the mapping is between 1 minute and 1 second.

e. System Environments for the Experiment

In order to parallelize the execution and speed up the process of getting the results, we ran the experiment for Postgres and MySql on the following environments:
1. PostgreSQL - Windows machine with Intel core i7 - 16 GB RAM
2. MySQL- MacOS machine with core Intel i5 - 8 GB RAM

We found the performance to be better on the MacOS machine which might have contributed to better performance of transaction execution in MySQL. Another influencing factor for the performance was that no other programs could be run on the machine which affected the performance of the experiment.

2. <u>Experiments performed along with results and analysis</u>

The experiments performed for Postgres and MySQL were done on two machines with different processing configurations due to the large amount of processing time for each experiment. The same set of operations are executed at different isolation levels.

1. Read Uncommitted

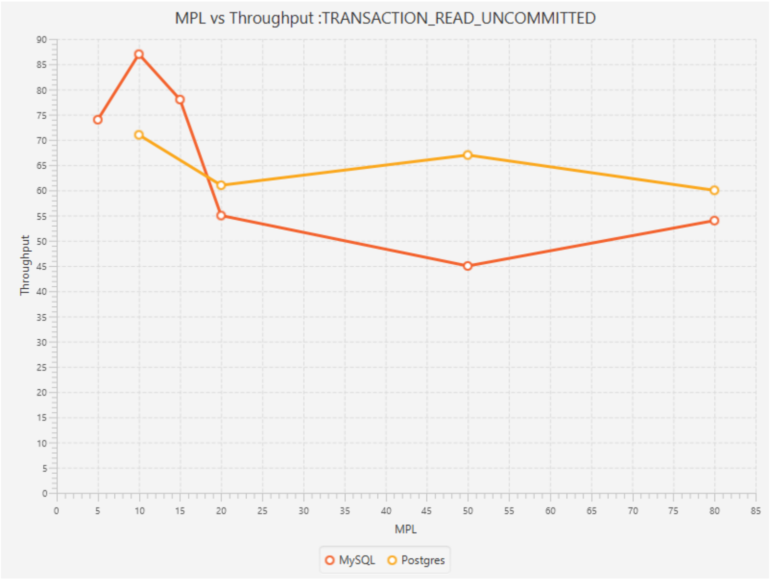
Read Uncommitted allows dirty read, non-repeatable read and phantom read. Initially, as the MPL increased, increase in throughput was observed. However, after that point, the throughput drastically reduced and then became steady. Phantom reads are allowed.

Analysis:

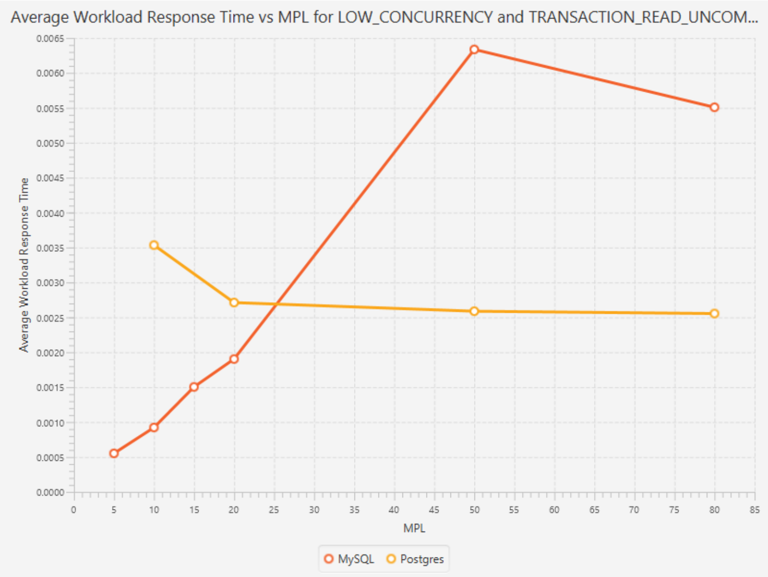
In case of Postgres, drastic change in throughput is not observed. As the level on MPL increased, so did the response time. Postgres maintained a near constant response time for the entire workload. Response time for queries i.e. read operations is observed to be less, since it is not restricted to reading committed data. High concurrency results show that the throughput is reduced, and the read response time is low.

a. Low concurrency:

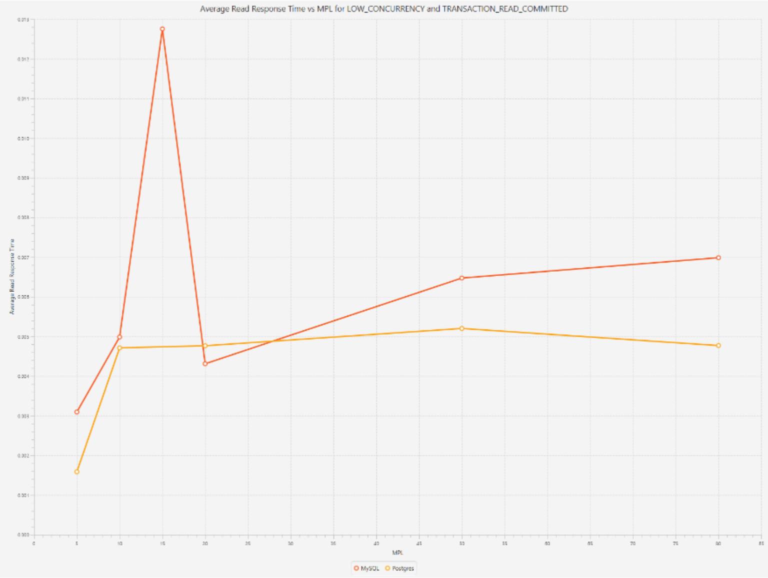
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL

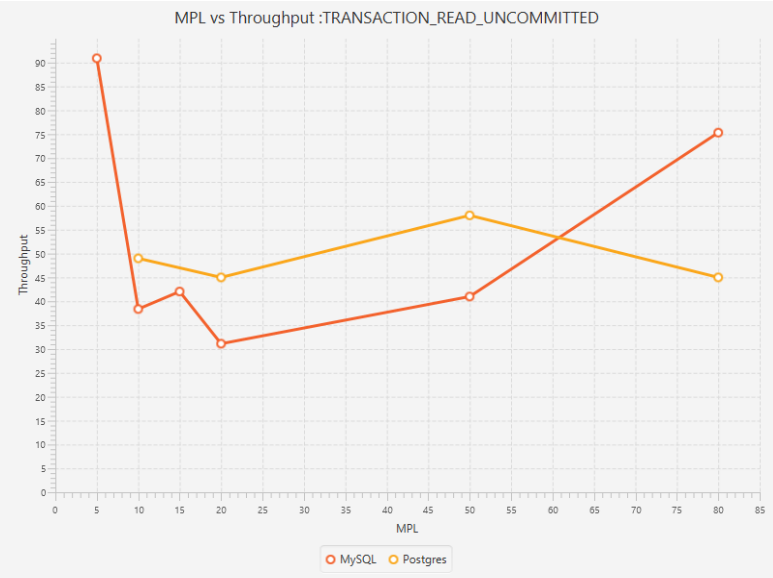


iii. Average Read Response Time vs MPL

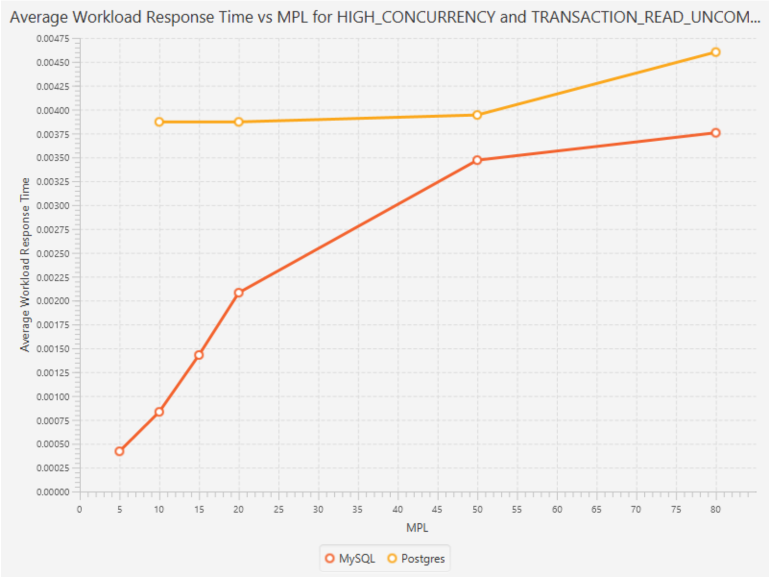


b. High concurrency:

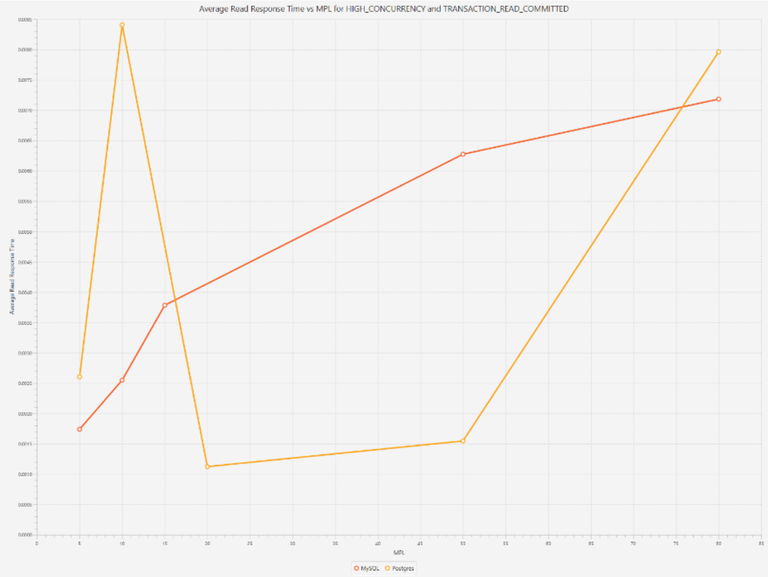
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL



iii. Average Read Response Time vs MPL



2. Read Committed

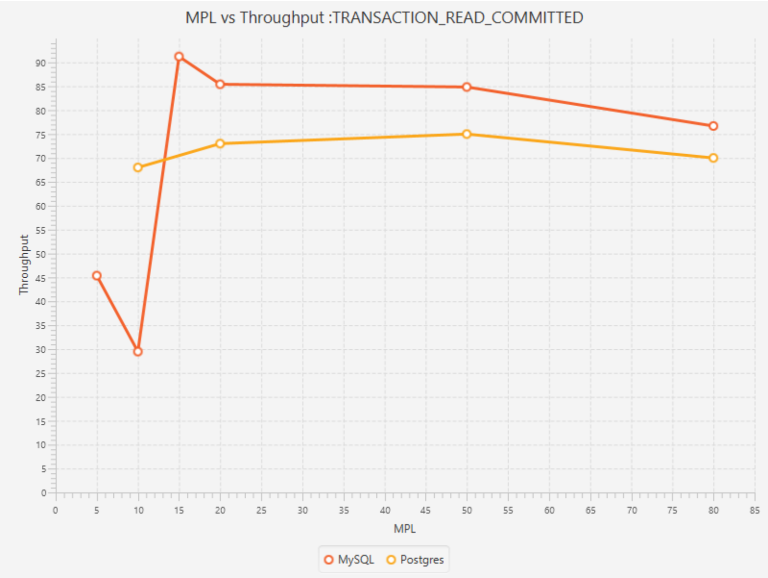
Read Committed allows the transactions to access only committed data to be read. It is the default isolation level in Postgres. Phantom reads are allowed.

Analysis:

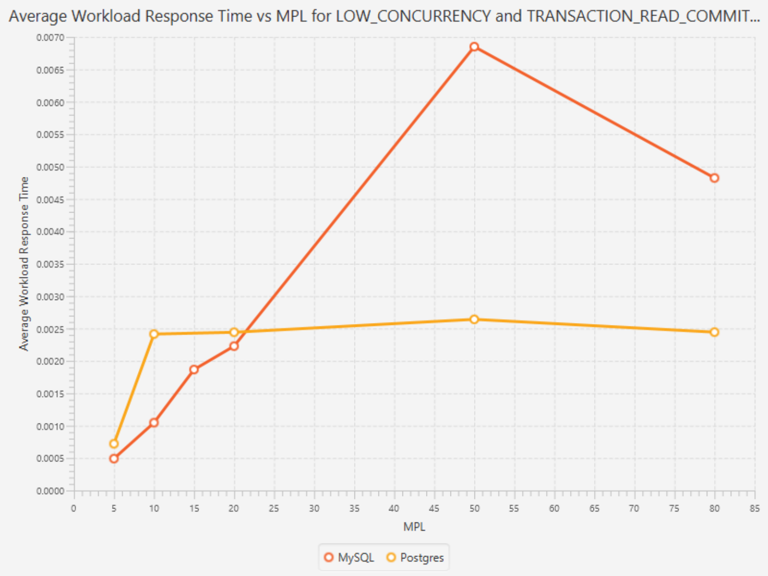
Throughput of Postgres at this isolation level is steady. MySQL shows a steady increase in the throughput which stabilizes after a point. The transaction sees the data that is committed before the transaction began. It does not check whether the data is updated or new data is inserted. Thus resulting in high performance. Response time increased as multiple transactions began writing the data. Select queries could be one reason for this increase, as the isolation level restricts the operations to read only committed data. Postgres shows a higher response time for the entire workload, however for select queries, it is lower than that of MySQL. As in the previous isolation level, the response time for MySQL for the entire workload increased linearly, while that of Postgres remained steady.

a. Low concurrency

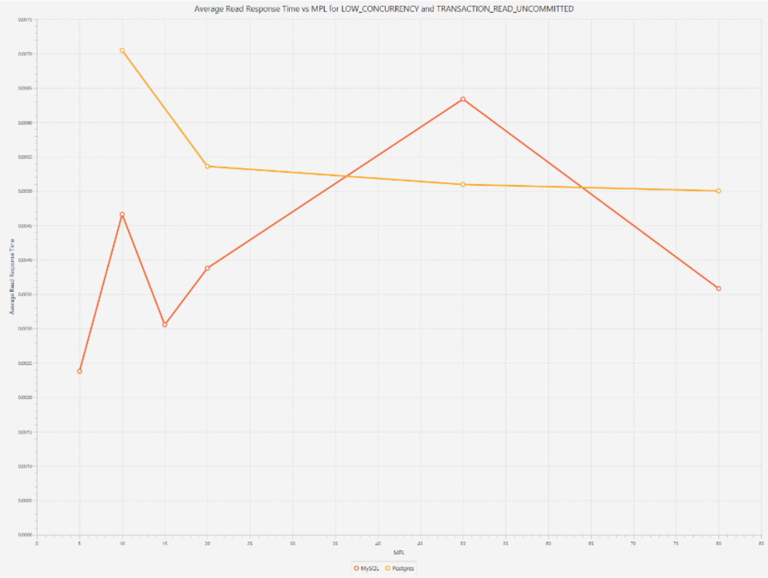
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL

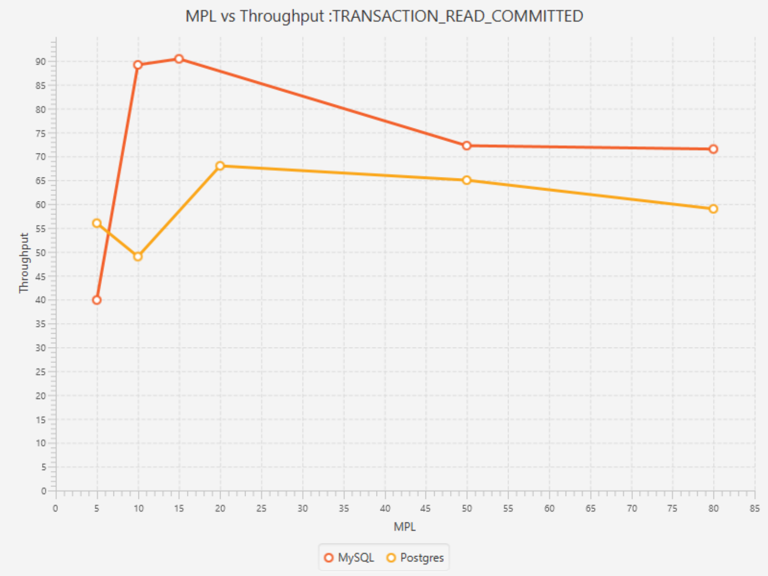


iii. Average Read Response Time vs MPL

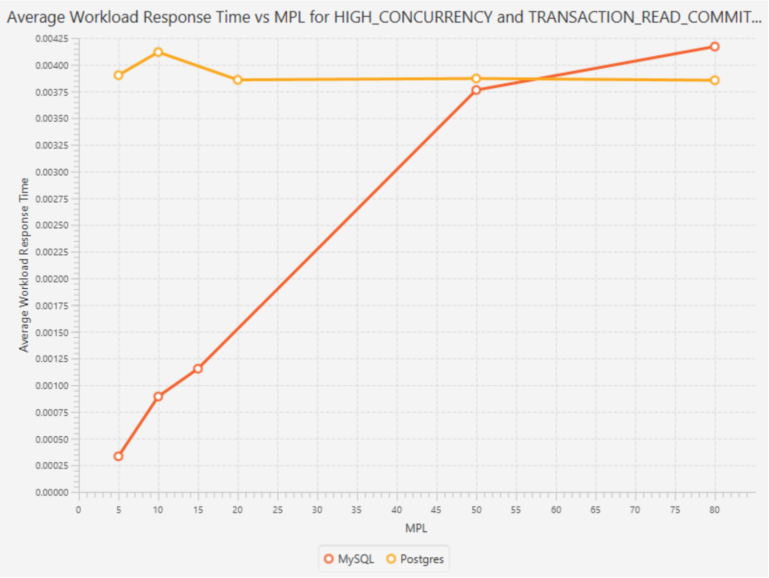


b. High concurrency

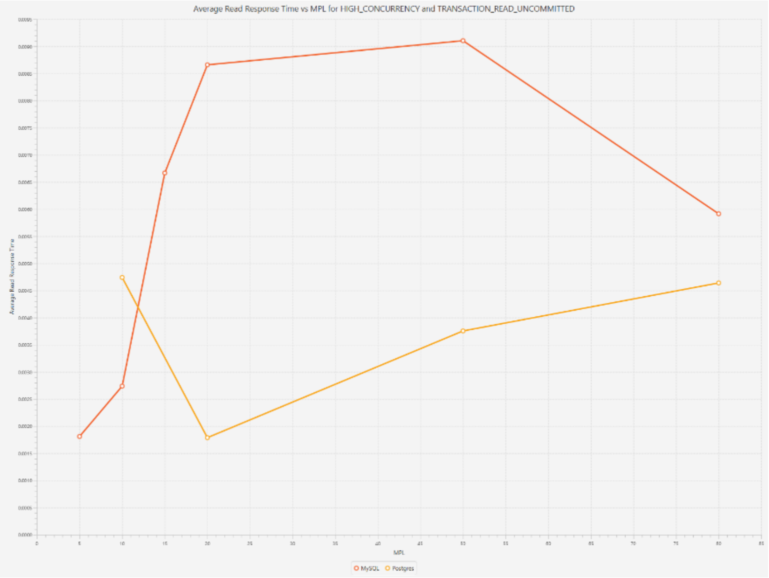
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL



iii. Average Read Response Time vs MPL



3. Repeatable Read

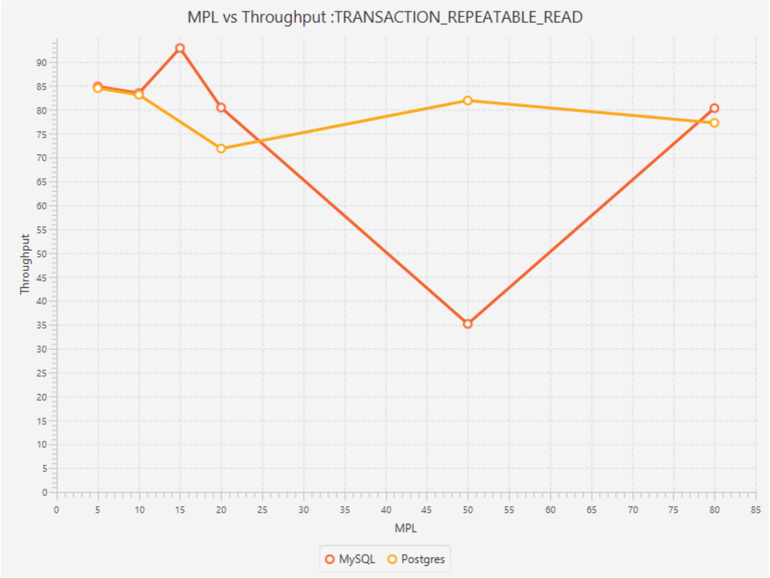
Repeatable read is a higher isolation level which along with read committed guarantees that the read data does not change and if the transaction were to read the data again, the value would be the same as before. It is the default isolation level for MySQL. Phantom reads are allowed.

Analysis:

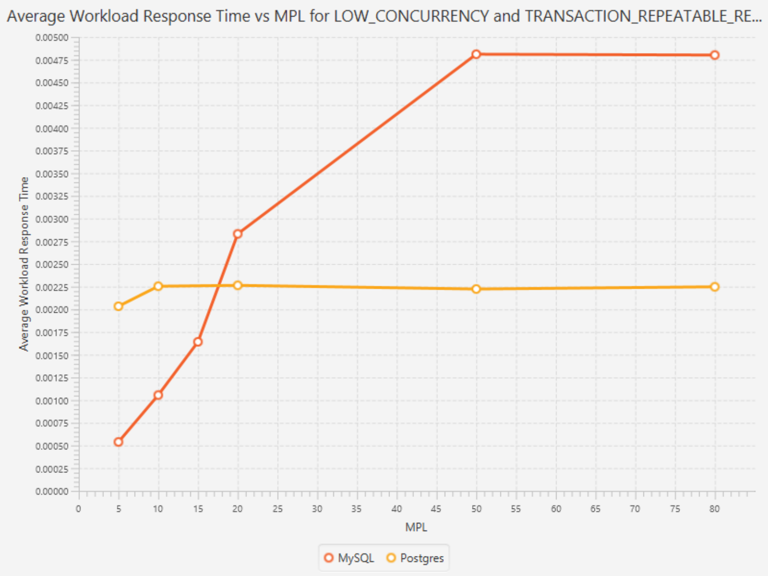
Operations performed in a transaction are inserts and selects. The isolation level affects those operations which update values read by the transaction. Since, that is not the case here, the throughput is high as no update transaction is blocked by transactions. Transactions implementing repeatable read work on the snapshot taken before execution of every transaction. Thus, the response time for workload and read queries is low for Postgres. Some spikes in the response time may be due to other processes executing on the test environment. Response time is high for high concurrency.

a. Low concurrency

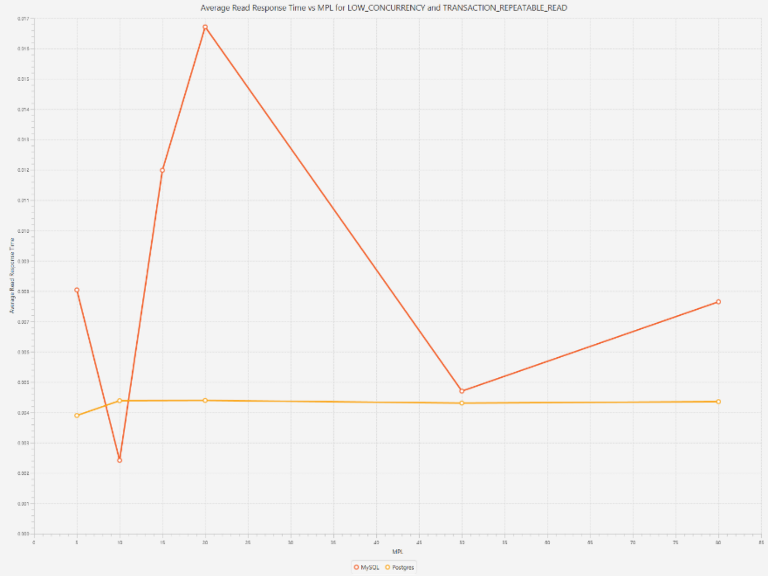
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL

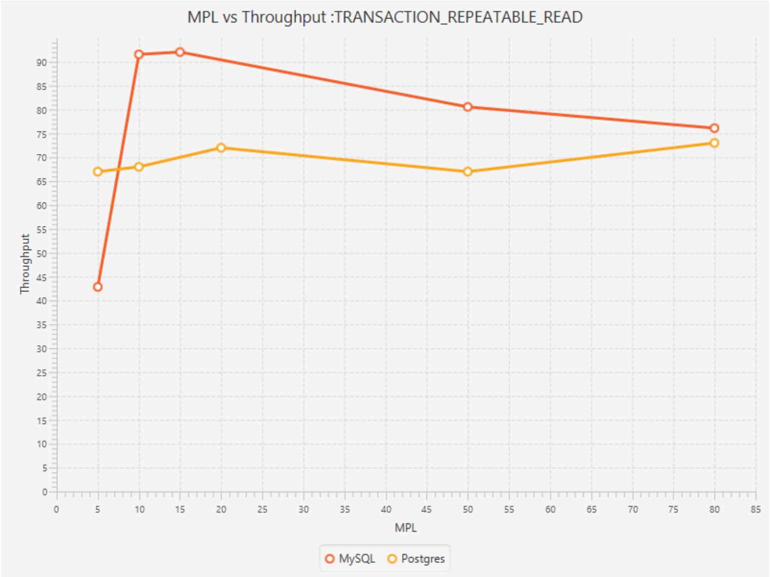


iii. Average Read Response Time vs MP

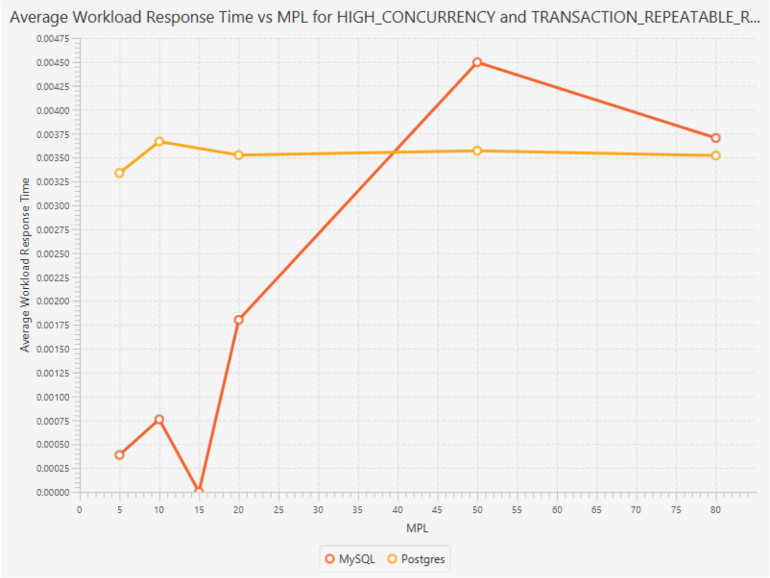


b. High concurrency

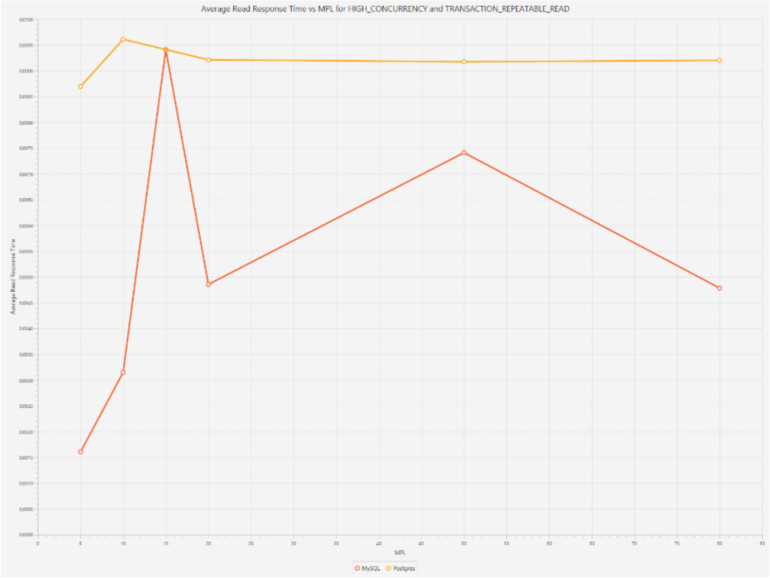
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL



iii. Average Read Response Time vs MPL



4. Transaction Serializable

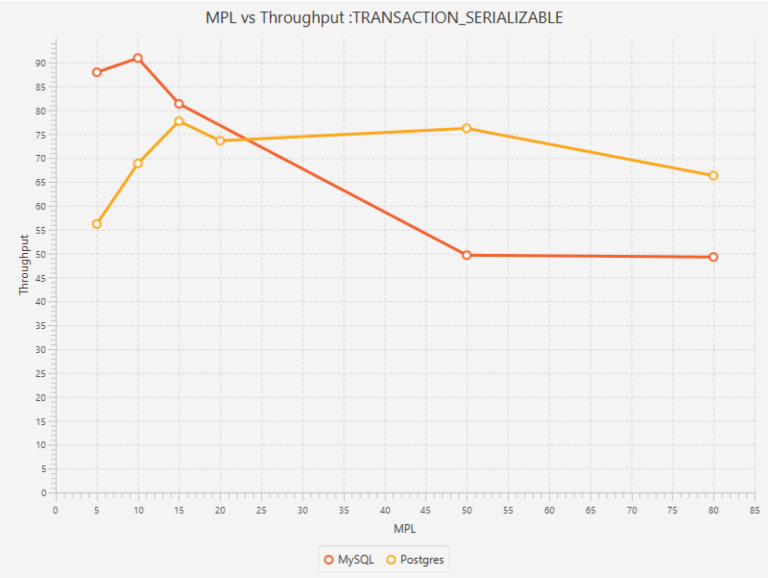
This isolation mode is the most strict isolation level. The transactions are executed in a serializable way. Along with enforcing the rules of the above three isolation levels, the additional constraint enforced is that phantom reads are not allowed. One key issue encountered while executing these transactions is the conflicts while executing transactions through multiple threads. The way we solved this issue was by retrying the aborted transactions.

Analysis:

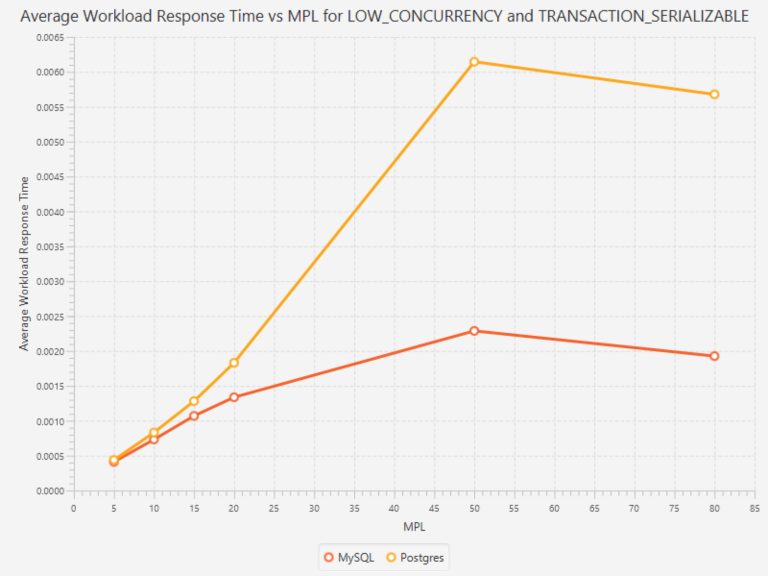
It is expected to have minimum throughput. As shown in the graph, throughput of MySQL drastically reduces after witnessing a peak in the throughput. On the other hand, for Postgres, the performance has improved with the level of MPL. The performance also depends on the grouping of the operations. If all inserts are grouped together and no select operation is running, then higher performance is observed. As seen in the previous results, the read response time of MySQL is higher than that of Postgres. In high concurrency, the response time for read queries and workload is quite similar.

a. Low concurrency

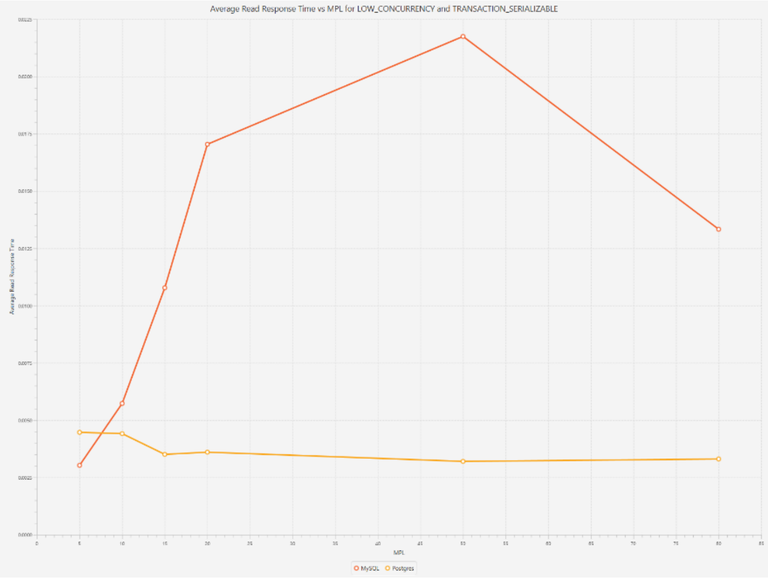
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL

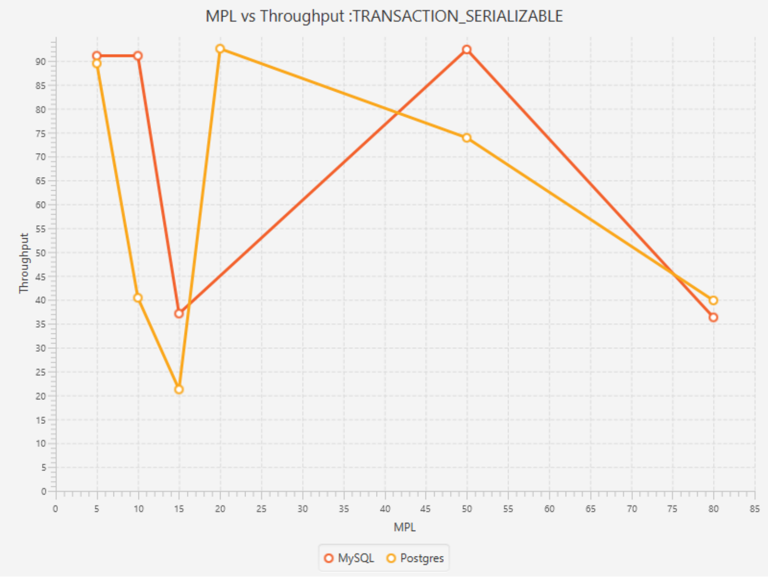


iii. Average Read Response Time vs MPL

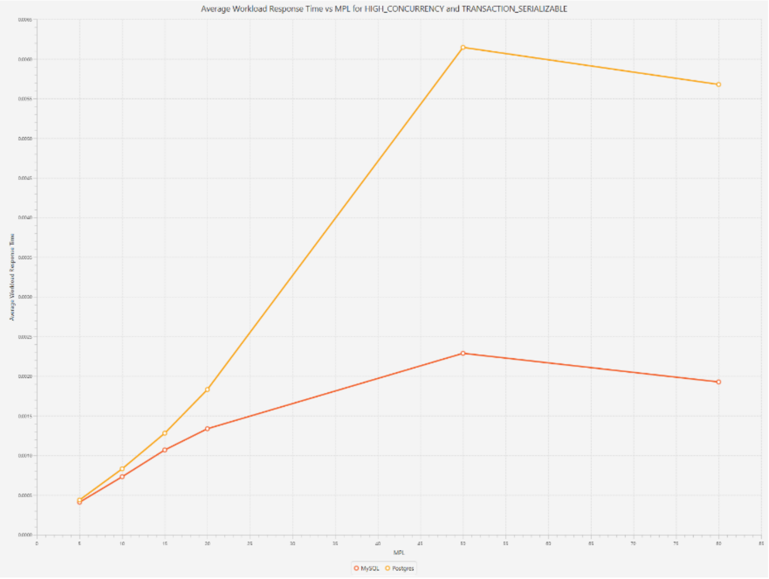


b. High concurrency

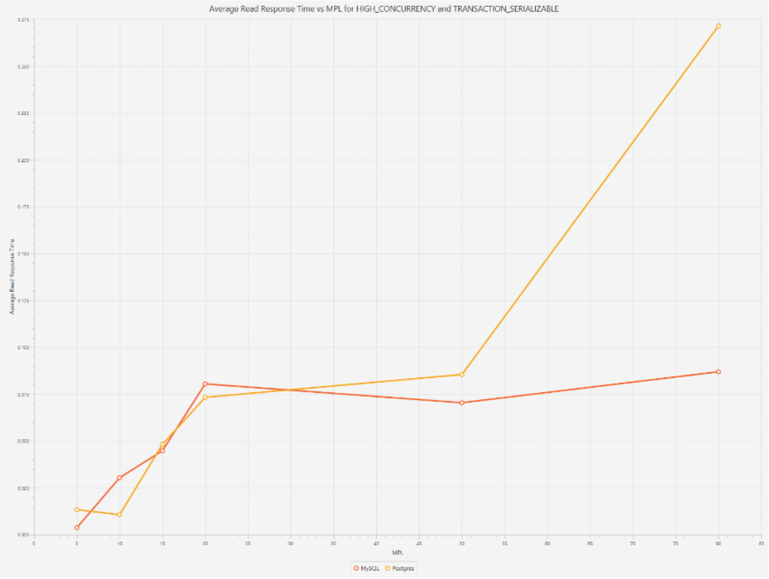
i. Throughput vs MPL



ii. Average Workload Response Time vs MPL



iii. Average Read Response Time vs MPL



Conclusion:

The performance of transaction execution varies for different levels of isolation for different concurrency levels. Through all the results, the common conclusion to draw is that the simulator performed the best when the MPL ranged from 10 to 20. The numbers for throughput and response time were the best in this range of MPL.