CS 223

**PROJECT 1**

**REPORT**

Utkarsha Ganla - 61447048

Tushar Kulkarni - 20648620

Arundhati Sawant - 29525453

1. **Design of the simulator**
   1. **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.
   1. **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.

* 1. **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.

* 1. **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.

1. **Experiments performed along with results and analysis**

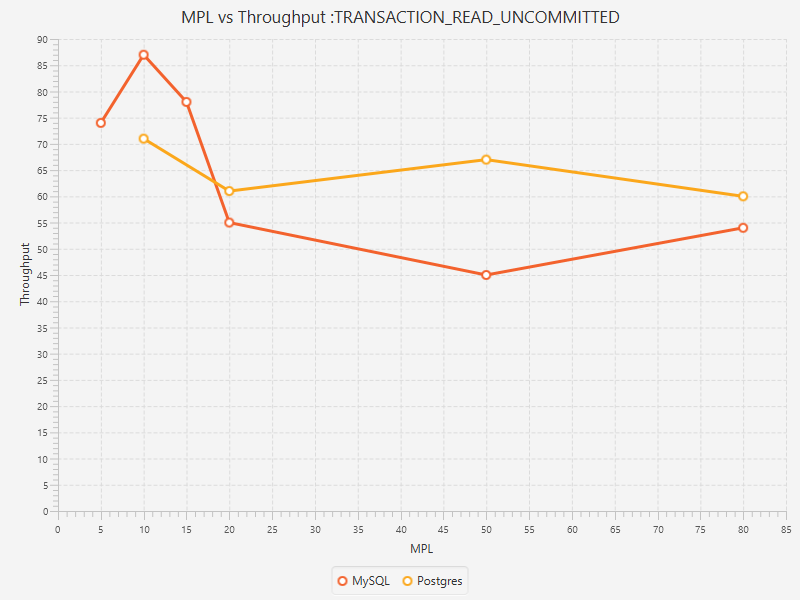
Throughput vs MPL

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

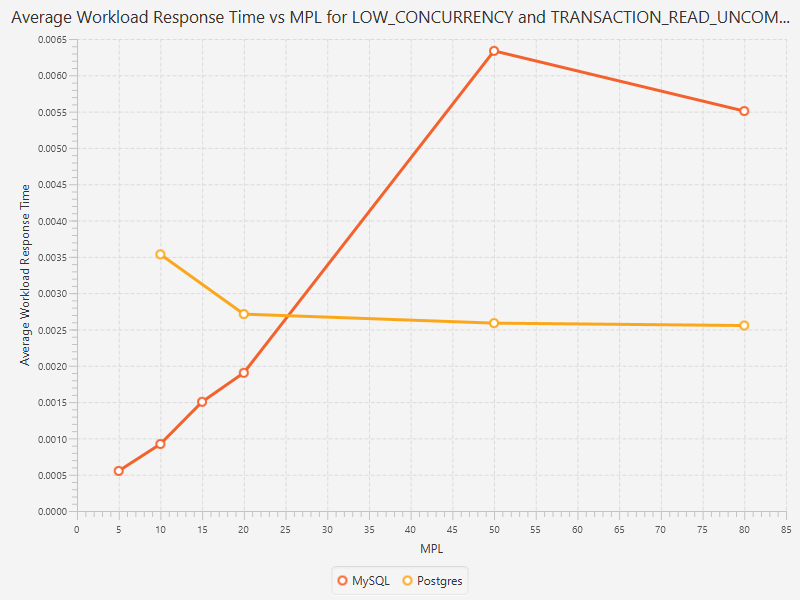
1. Read Uncommitted

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. 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.

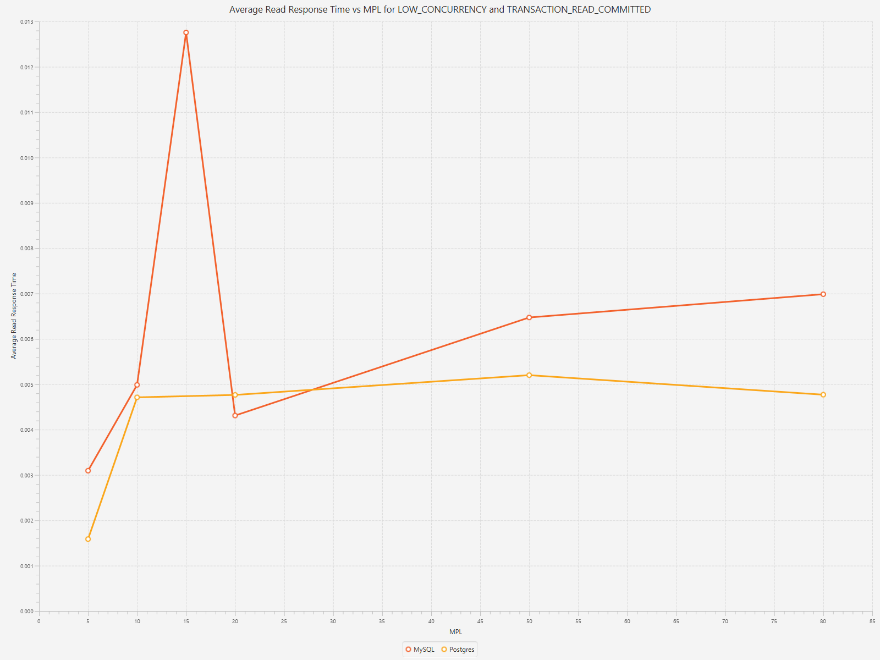
* 1. Low concurrency:
     1. Throughput vs MPL



* + 1. Average Workload Response Time vs MPL

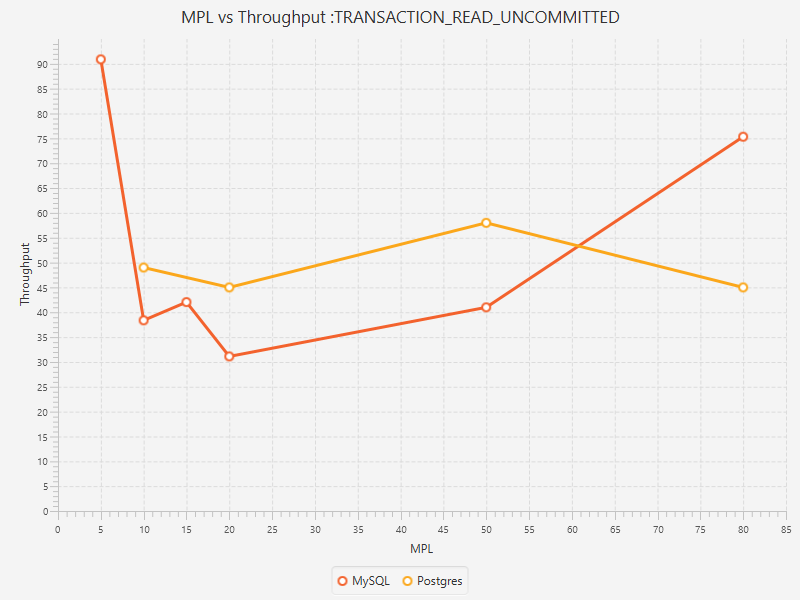


* + 1. Average Read Response Time vs MPL

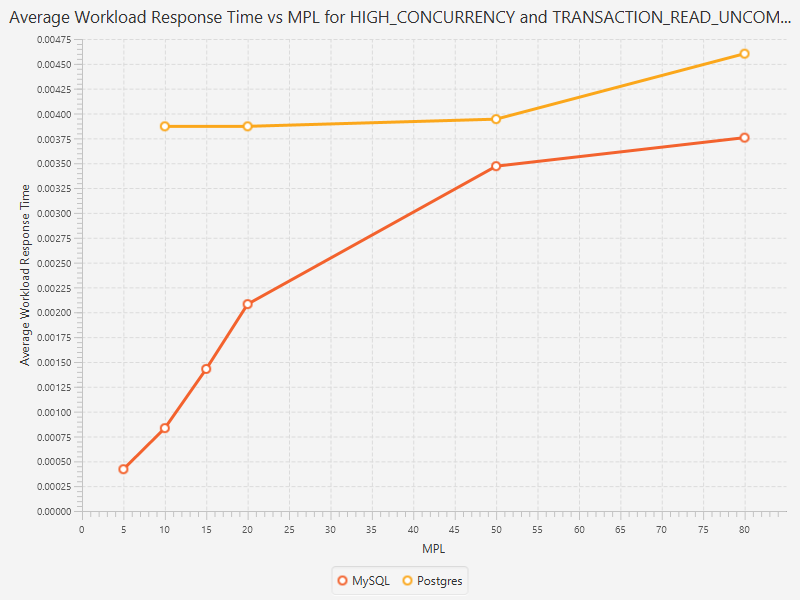


* 1. High concurrency:

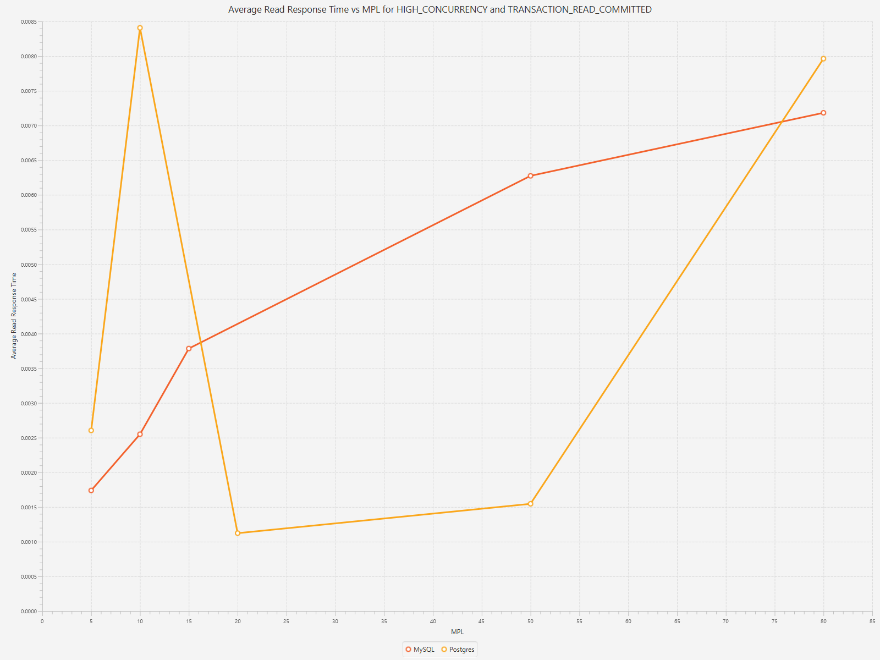
Throughput vs MPL



Average Workload Response Time vs MPL



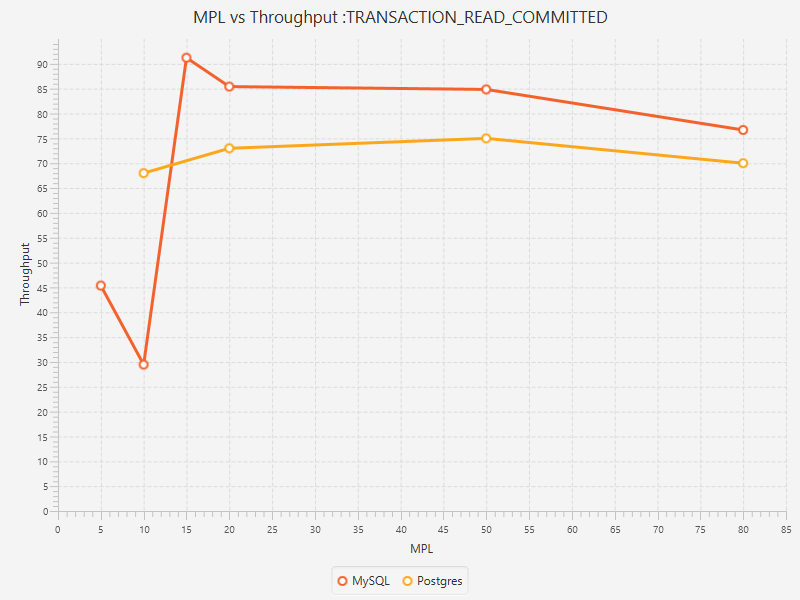
Average Read Response Time vs MPL



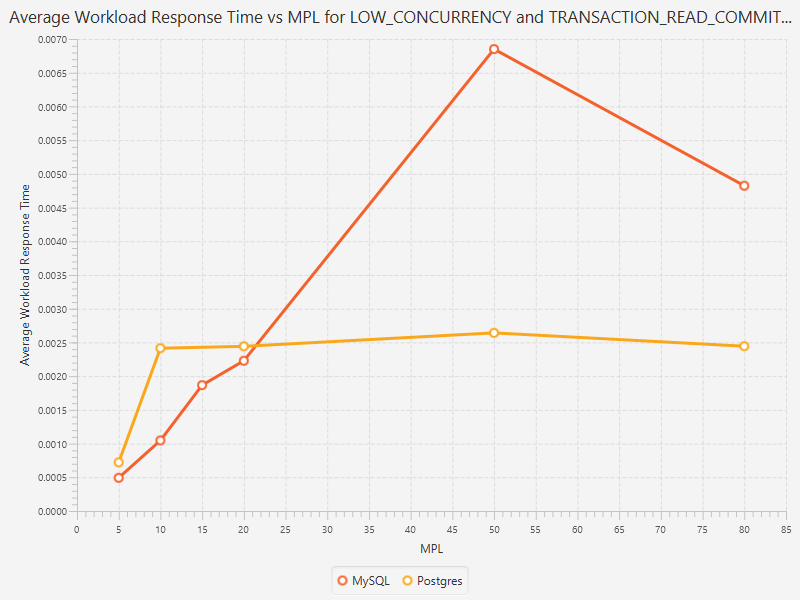
1. Read Committed

Read Committed is the default isolation level in Postgres. 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.

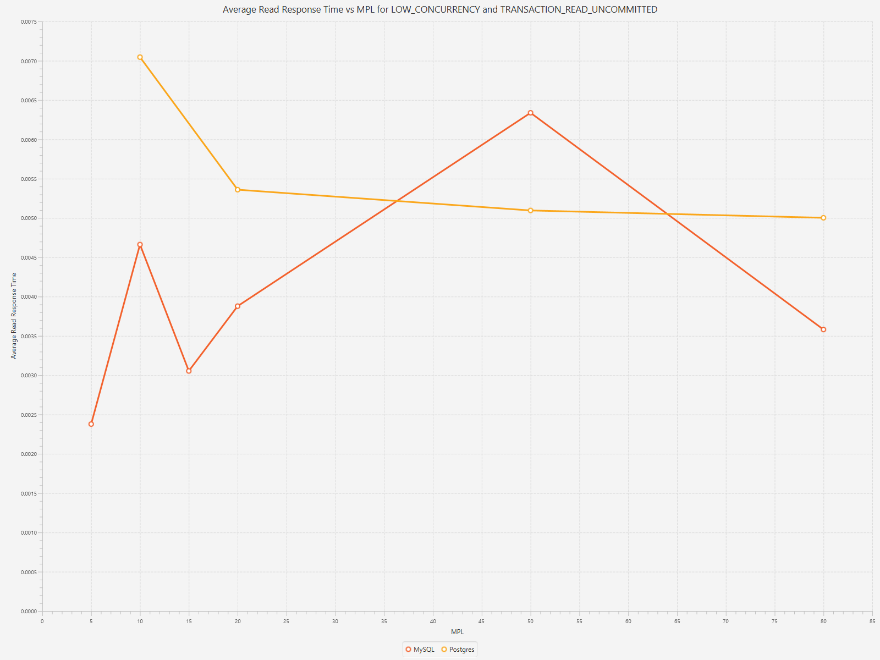
* 1. Low concurrency
     1. Throughput vs MPL



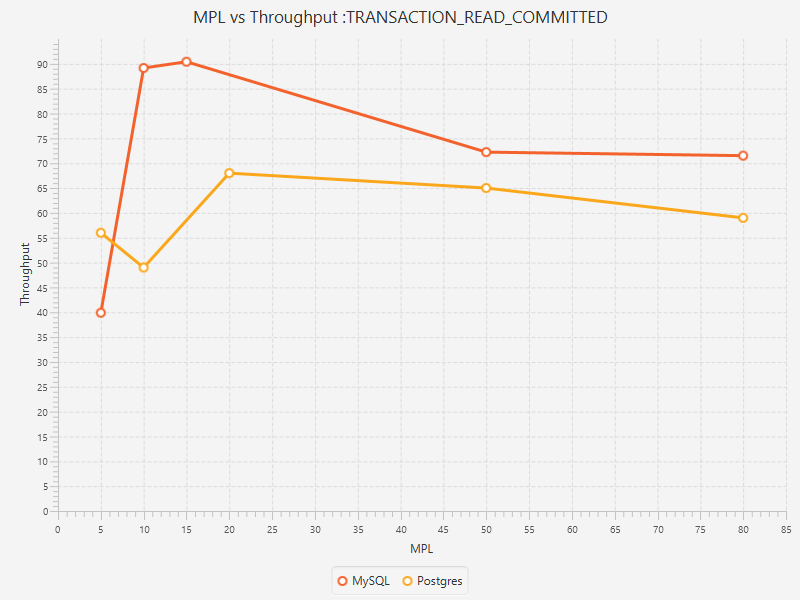
* + 1. Average Workload Response Time vs MPL



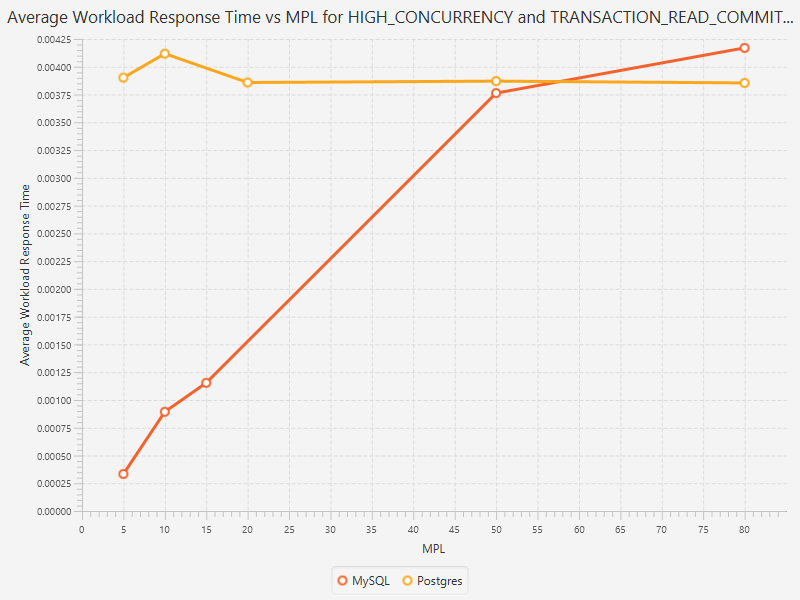
* + 1. Average Read Response Time vs MPL



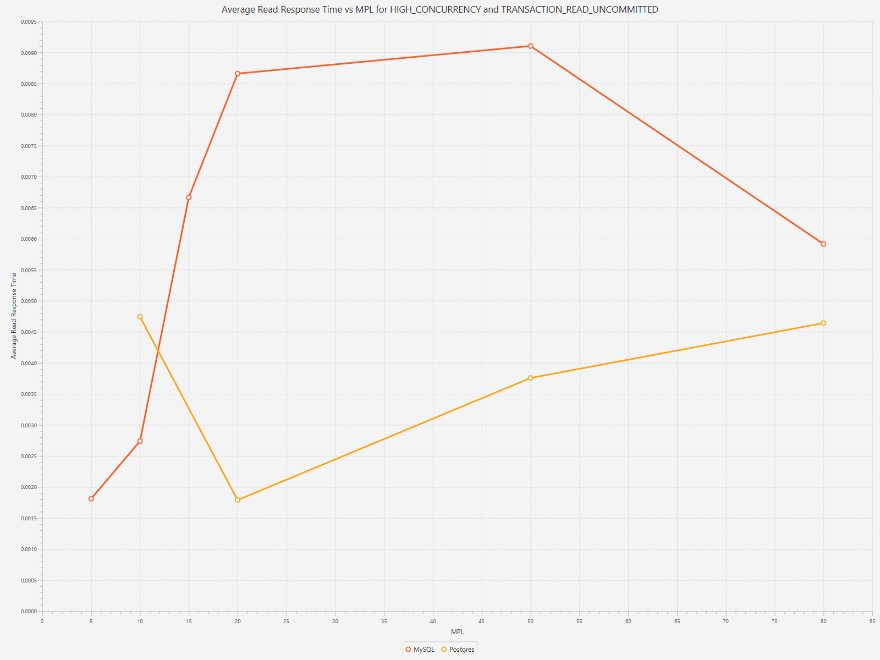
* 1. High concurrency
     1. Throughput vs MPL



* + 1. Average Workload Response Time vs MPL



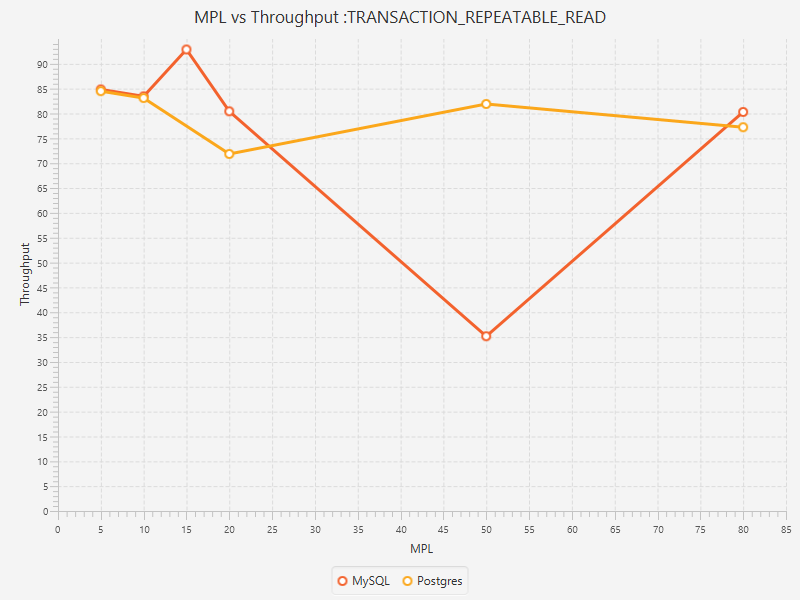
* + 1. Average Read Response Time vs MPL



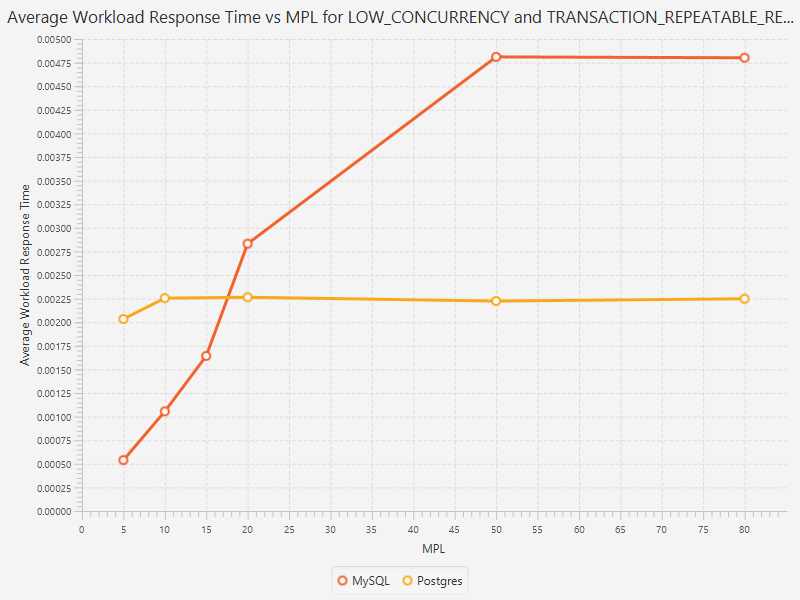
1. Repeatable Read

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. It is the default isolation level for MySQL. 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.

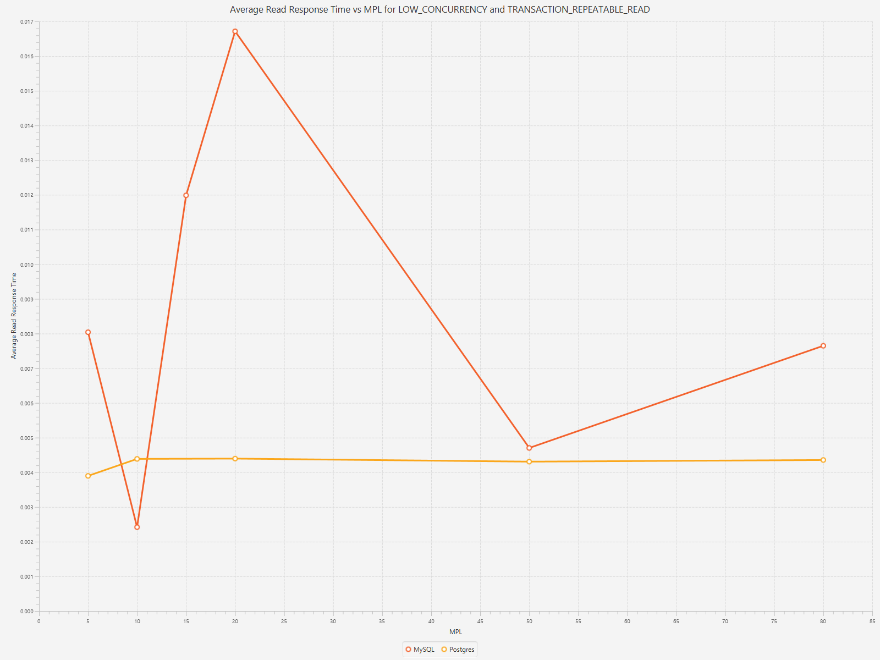
* 1. Low concurrency
     1. Throughput vs MPL



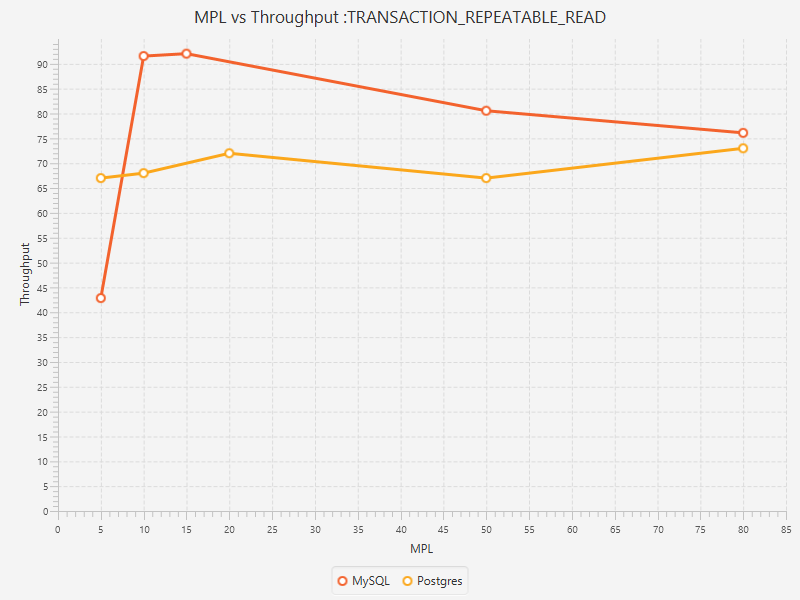
* + 1. Average Workload Response Time vs MPL



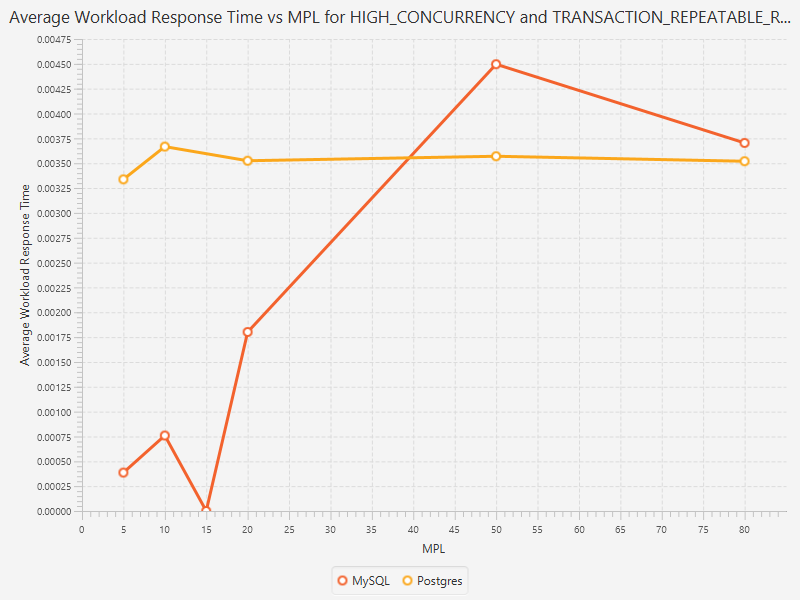
* + 1. Average Read Response Time vs MPL



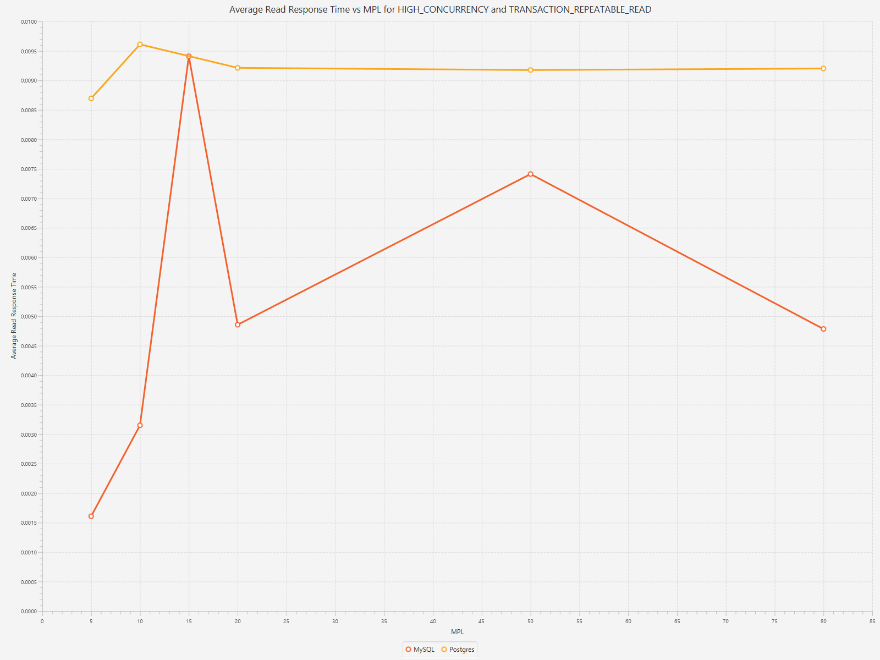
* 1. High concurrency
     1. Throughput vs MPL



* + 1. Average Workload Response Time vs MPL



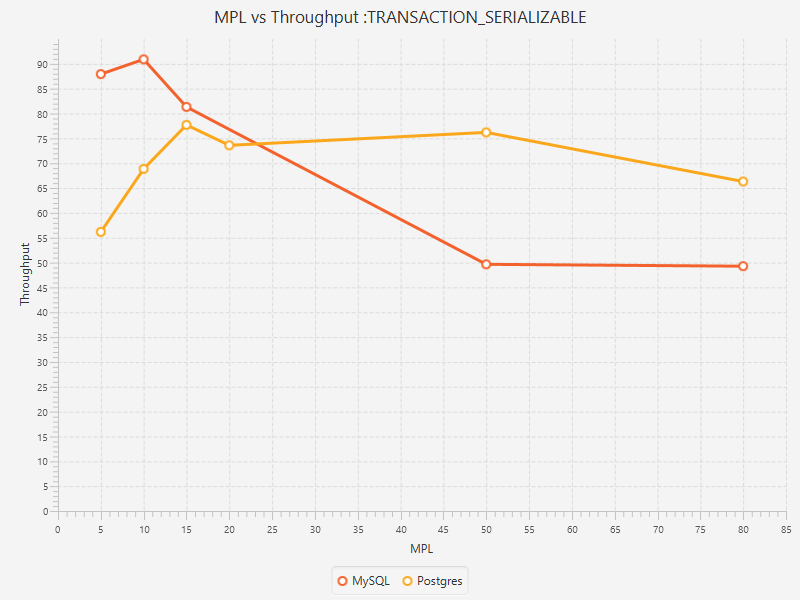
* + 1. Average Read Response Time vs MPL



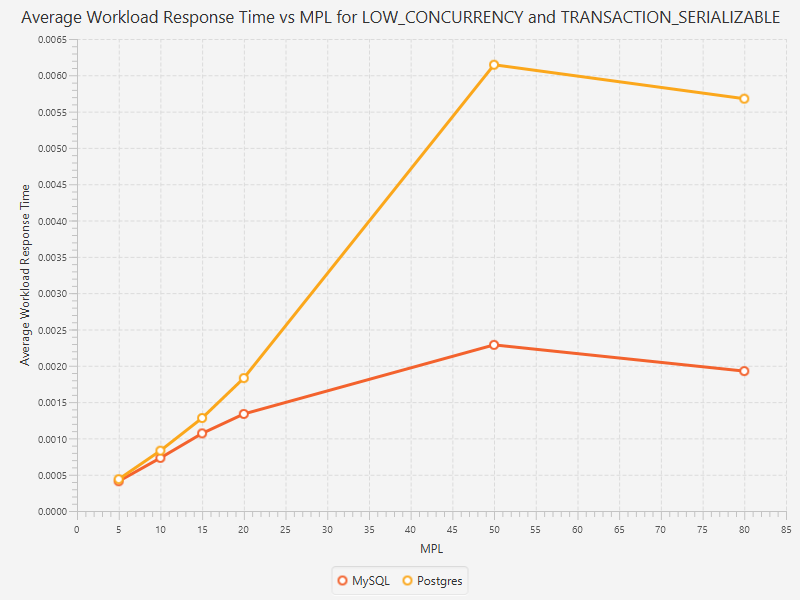
1. Serializable

This isolation mode is the strictest isolation level. 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.

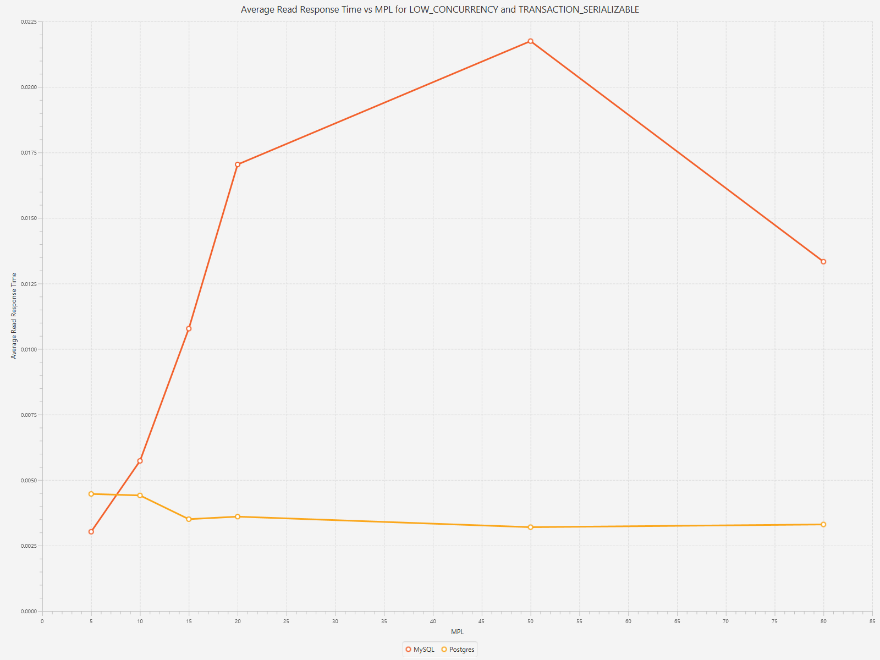
* 1. Low concurrency
     1. Throughput vs MPL



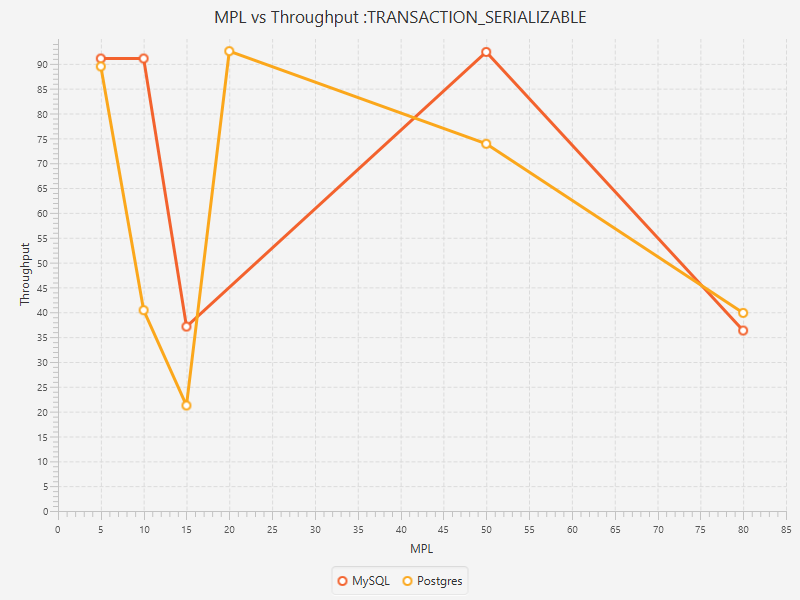
* + 1. Average Workload Response Time vs MPL



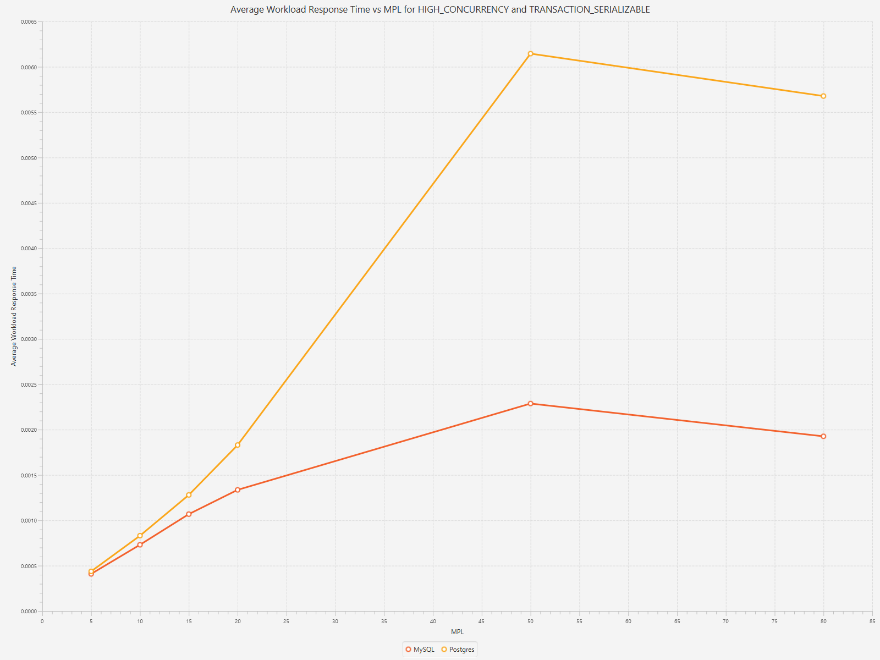
* + 1. Average Read Response Time vs MPL



* 1. High concurrency
     1. Throughput vs MPL



* + 1. Average Workload Response Time vs MPL



* + 1. Average Read Response Time vs MPL

