**CS 223 Project Part 1 Report**

Group:

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 into a file

**Simulation**

The given set of queries spans over 20 days. To simulate the actual execution the 20 days has been mapped to 20 minutes. Therefore, all operations happening in 3 minutes are mapped to one second. 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.

**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.

**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.

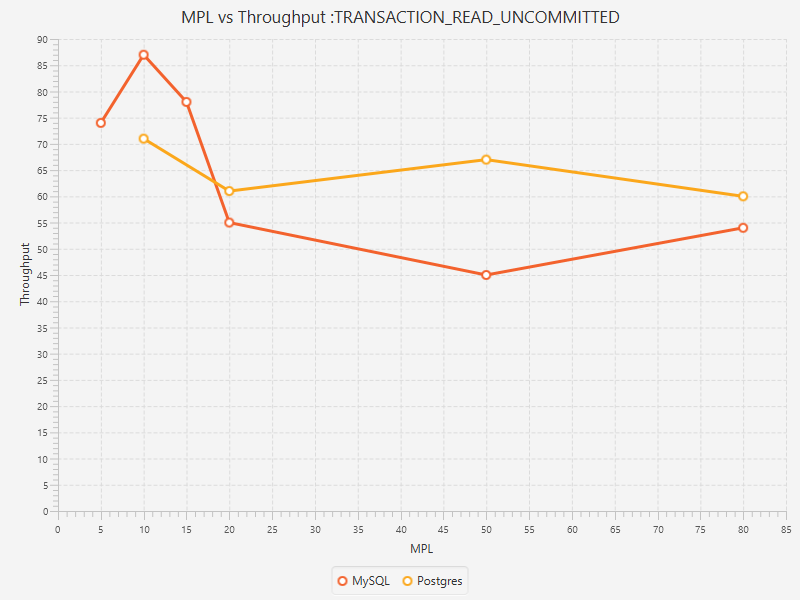
* 1. **Experiments performed along with results and analysis**

Throughput vs MPL - Low concurrency

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.

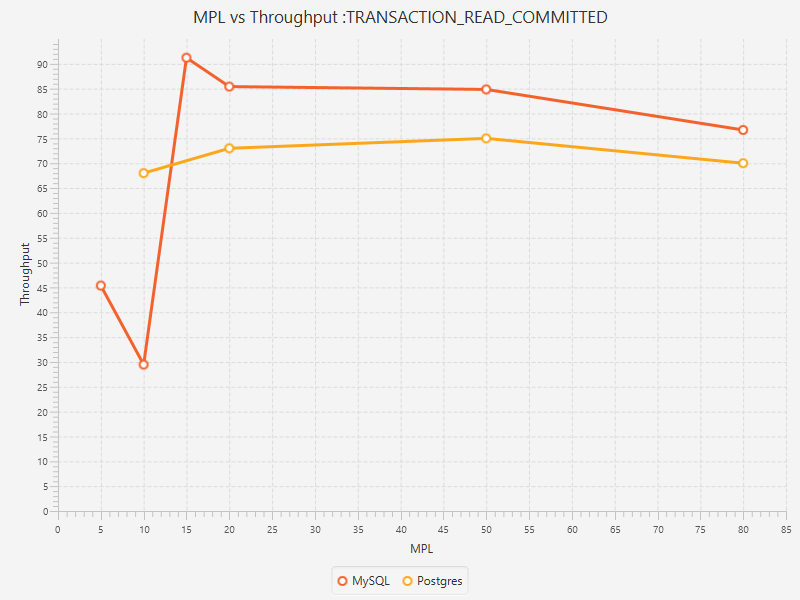
1. Transaction Read Uncommitted

The transaction 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. Allowing multiple threads to execute the transactions concurrently and allowing them to read uncommitted data leads to many rollbacks, reducing the throughput.



2. Transaction read committed

Read committed is 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.



High concurrency