Assignment 1

CSCI 729 NoSQL and NewSQL Databases

-Sidhartha Amperayani

Performance

1 thread:
Number of operations: 1346996
Products with stock level less than zero: 0

2 threads:
Number of operations: 731323
Products with stock level less than zero: 0

3 threads:
Number of operations: 965403
Products with stock level less than zero: 0

4 threads:
Number of operations: 1409761
Products with stock level less than zero: 0

5 threads:
Number of operations: 1387932
Products with stock level less than zero: 0

6 threads:
Number of operations: 1498876
Products with stock level less than zero: 0

7 threads:
Number of operations: 1621732
Products with stock level less than zero: 0

8 threads:
Number of operations: 1488759
Products with stock level less than zero: 0

9 threads:
Number of operations: 1245142
Products with stock level less than zero: 0

10 threads:
Number of operations: 1541014
Products with stock level less than zero: 0

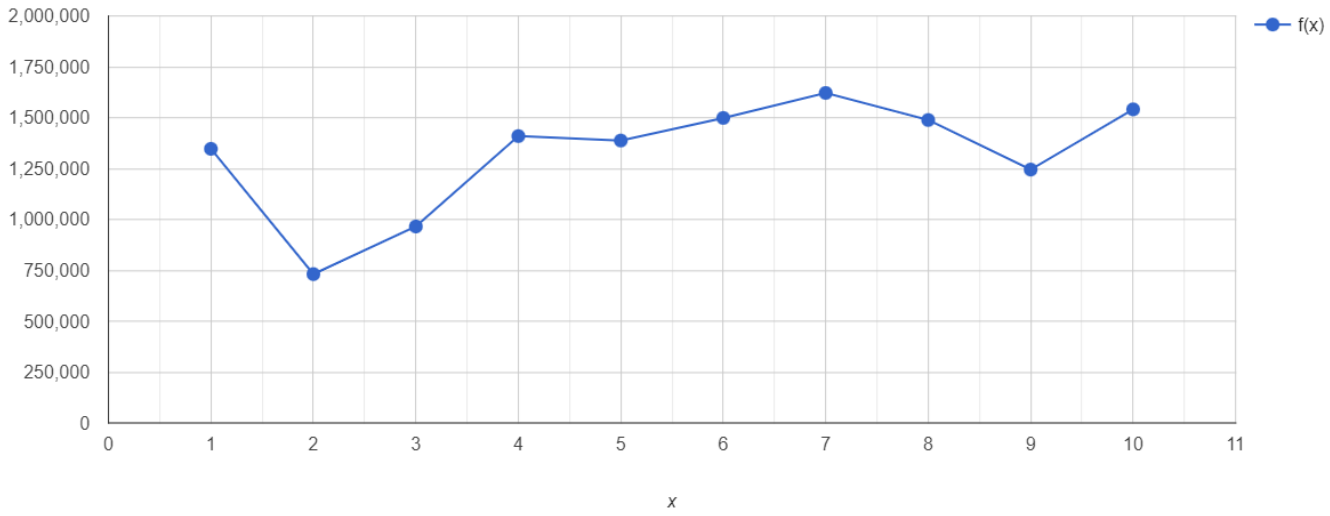

Figure1 X-axis: Number of threads; Y-axis: Total number of Operations

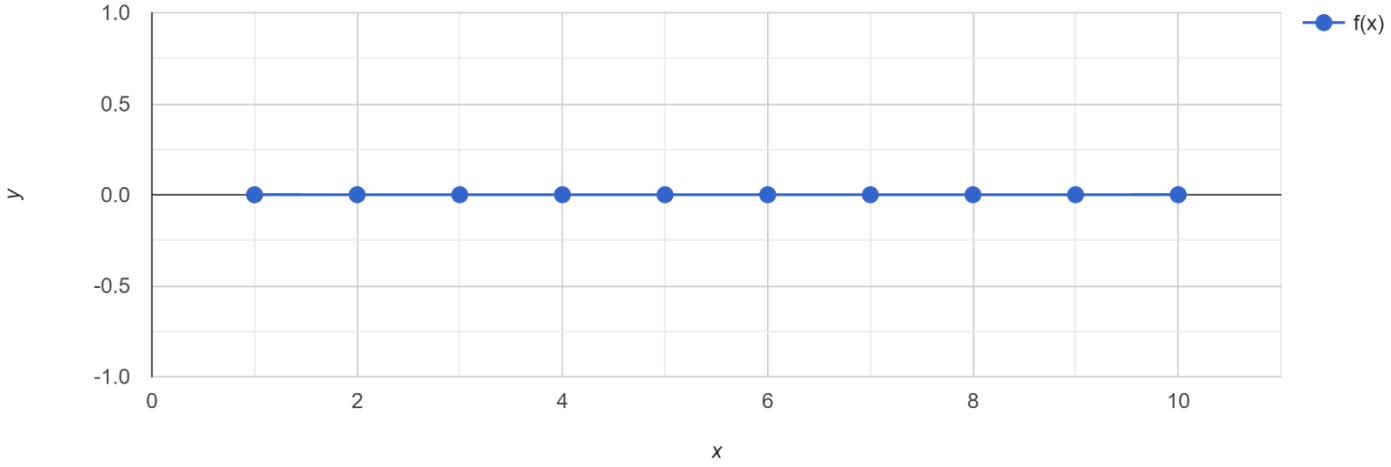

Figure 2 X-axis: Number of threads; Y-axis: Percentage of products with negative stock level

Conclusions

- Serializable ensures that the threads running concurrently wouldn't counter-effect the database. Because of the constraints on 'post review' and 'submit order' and the Transaction Isolation set to TRANSACTION_SERIALIZABLE the product stock level is not going under 0.
- Serialization may become a potential performance issue, in a heavy load scenario such as this, hence one of the reasons for a non–uniform curve.
- Using multithreading the resultset of a query can be shared among multiple cores of the system. This feature might or might not help improving the overall

performance. This is another reason why parallelism's performance when compared to single thread is both stunted and shows considerable improvement in performance.

SQL Schema and Code

Please find the *SQLSchema.txt* file

File Details

1. *Transactions_729.java* file is where initialization/populating the tables is done and all 7 operations are implemented.
2. *threadsTransaction.java* file has the threads which run concurrently on all 7 operations on the basis of probability.
3. Run the latter file.