Programming Assignment 2: Implementing BTO and MVTO algorithms

Goal: The goal of this assignment is to implement the BTO and MVTO algorithms studied in class in C++.

Implementation Details:

File input variables:
No. of threads(nThreads) = 10, 20, 30, 40, 50
No. of variables(m) = 10
constVal = 100
Lambda = 20

Data Structures:
map<int,int> status -> for maintaing status of the transactions which enter the execution cycle.
map<int,vector<int> > wL -> for maintaing write list of all variables
map<int,int> maxRshd -> for maintaing most recent transaction reading each data item.
map<int,map<int,int> > v_Versions -> for maintaining the versions of each data item.
map<int,int> tx_log ->for logging transaction data locally

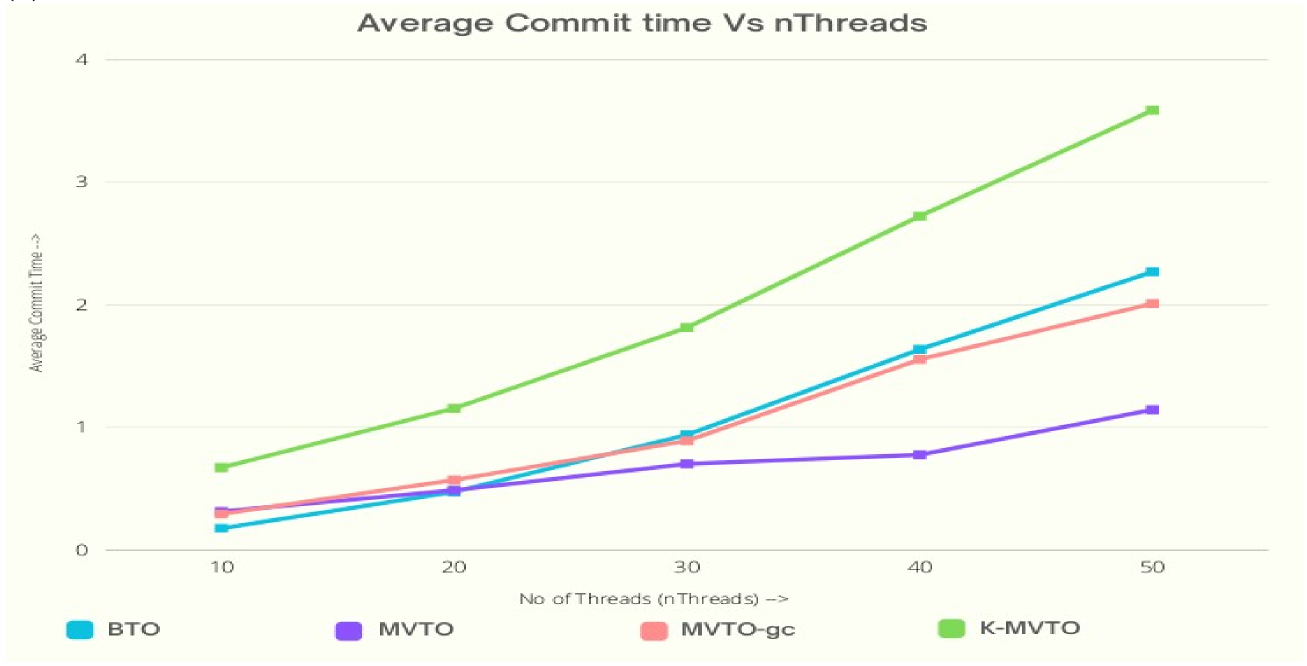
Mutex Locks:
vLock, pLock, mLock, cLock, idLock

For each operation:
begin_trans() = acquire idLock -> assign tx_id -> release idLock -> acquire vLock -> initialize status, tx_log -> release vLock
read(int tx_id,int dx) = acquire vLock -> checks the specified condition for updating local version to log, modify status, maxRshd as per the code -> release vLock
write(int tx_id,int dx) = acquire vLock -> check tx_id with maxRshd[dx] and wL[dx] size -> add to wL, local version to tx_log -> release vLock
cleanup(int tx_id) = acquire cLock -> remove tx and related data from v_Versions/tx_log as per the specified condition -> release cLock
tryCommit(int tx_id) = checks the conditions for commiting the transaction as specified with BTO, MVTO, MVTO-gc and K-MVTO.

Average commit time = commitTime/nThreads
Average time taken by tx to commit successfully starting from begin_trans().
Average abort count = abortCountGlobal/nThreads
Average no. of tx aborted per commited tx.

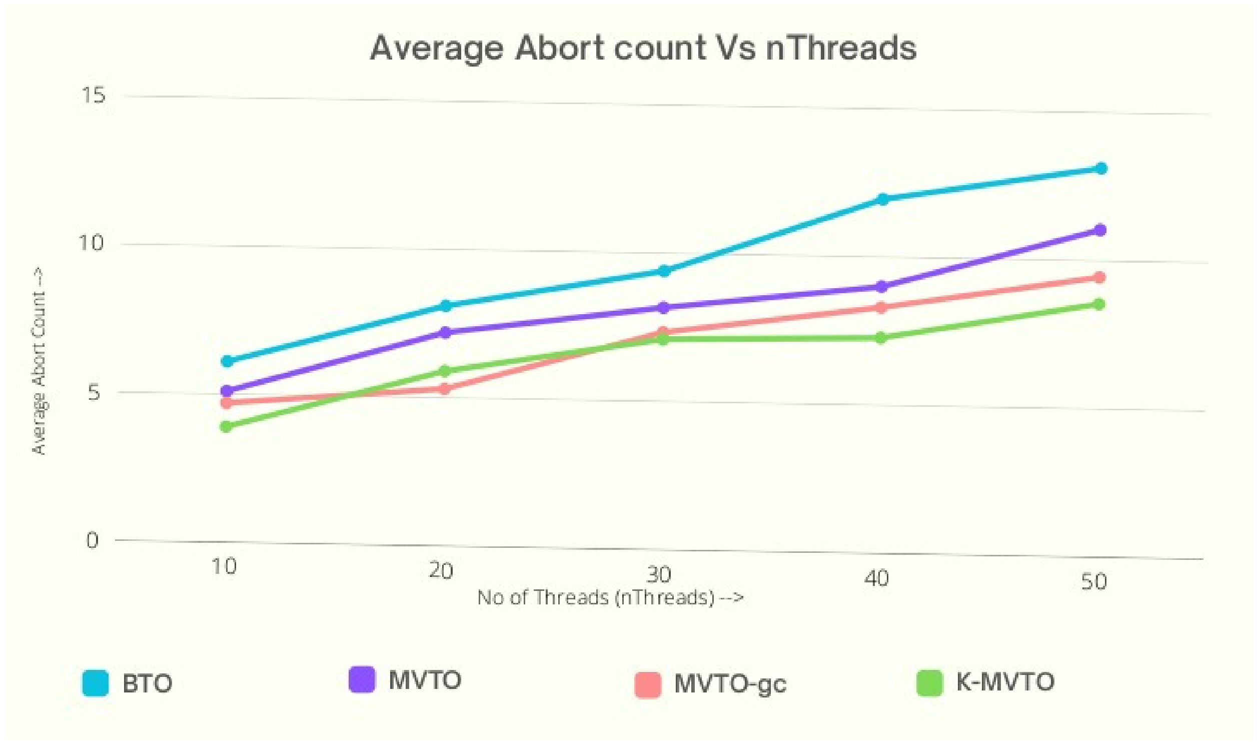
Graphs:

(1)



This graph plots Average delay to commit vs Total no. of threads.

(2)



This graph plots Average abort count vs Total no. of threads.

Observations:
The no. of aborts is higher in case of BTO than compared to MVTO. As the number of transactions increases, the abort count also increases, which in turn affects the commit time

(increases). This is because as more transactions are added, the contention for resources (such as locks) increases, which increases the likelihood of conflicts and therefore no. of aborts.
In contrast, in MVTO, the number of threads or transactions is not directly related to the abort count as each transaction is having its own set of data versions. Hence, abort count is lower than BTO. Also, among different versions of MVTO, we can see that K-MVTO has relatively lower abort count.
In MVTO, commit delay is lower because conflicts are resolved by creating new versions of the data rather than aborting transactions. Therefore, even if a transaction encounters a conflict, it can still commit relatively quickly, since it operates on its own version of the data.
Thus BTO has higher abort counts and commit delay as no. of transactions increases than compared to MVTO.