Aayush Kumar
CO21BTECH11002
Assignment 4

The OBS class:

The update method allows a thread to update the value at a specified index id in the shared array. It loads the current stampedValue at the given index, creates a new stampedValue with an updated value, an incremented stamp, and the thread ID, and stores this new value back into the array at the same index. The collect method is used to collect a snapshot of the values in the shared array. It creates a copy of the shared array and returns it as a vector of stampedValue elements. The scan method scans the shared array and returns the values from it. It continuously collects snapshots of the array until two consecutive snapshots yield the same result (i.e., the values are not changing). Once this condition is met, it extracts and returns the values from the last collected snapshot.

The WFS class:

The WFS class maintains a helpsnaps vector, which stores snapshots for each writer thread. These snapshots are used for helping other threads during scans. The update method allows a thread to update the value at a specified index in the shared array. After the update, it takes a snapshot and stores it in the helpsnaps vector for the current thread. The collect method is responsible for collecting a snapshot of the values in the shared array. It returns a vector of stampedValue elements representing the current state of the array. The scan method continuously scans the shared array to obtain a consistent snapshot. It compares consecutive snapshots and checks if they are equal. If they are then it returns the snapshot. If not, it looks for opportunities to get help from other threads that have moved. If no help is available, it continues scanning. Otherwise, it returns the snapshot from a helper thread.

writer function:

Used to simulate writer threads. The function generates random values and locations. It uses a random number generator initialized with the thread's ID and the current time. It uses two types of random distributions: exponential distribution for generating time intervals (t) and uniform distribution for selecting a location (l) and value (v). The function records the start time using the gettimeofday function and calculates the time it takes to complete an update. This duration is stored in microseconds and added to a vector (wrTime) for tracking write times. It invokes an mrmwSnapObj.update method to update the shared array with the random value and location. It creates a log message describing the write operation, including the thread ID, the value, the location, and the timestamp when the update occurred. This log message is stored in an array (logs) at a specific index. After completing a write operation, the writer thread sleeps for a random duration (t) determined by the exponential distribution. This introduces variability in the write frequency.

snapshot function:

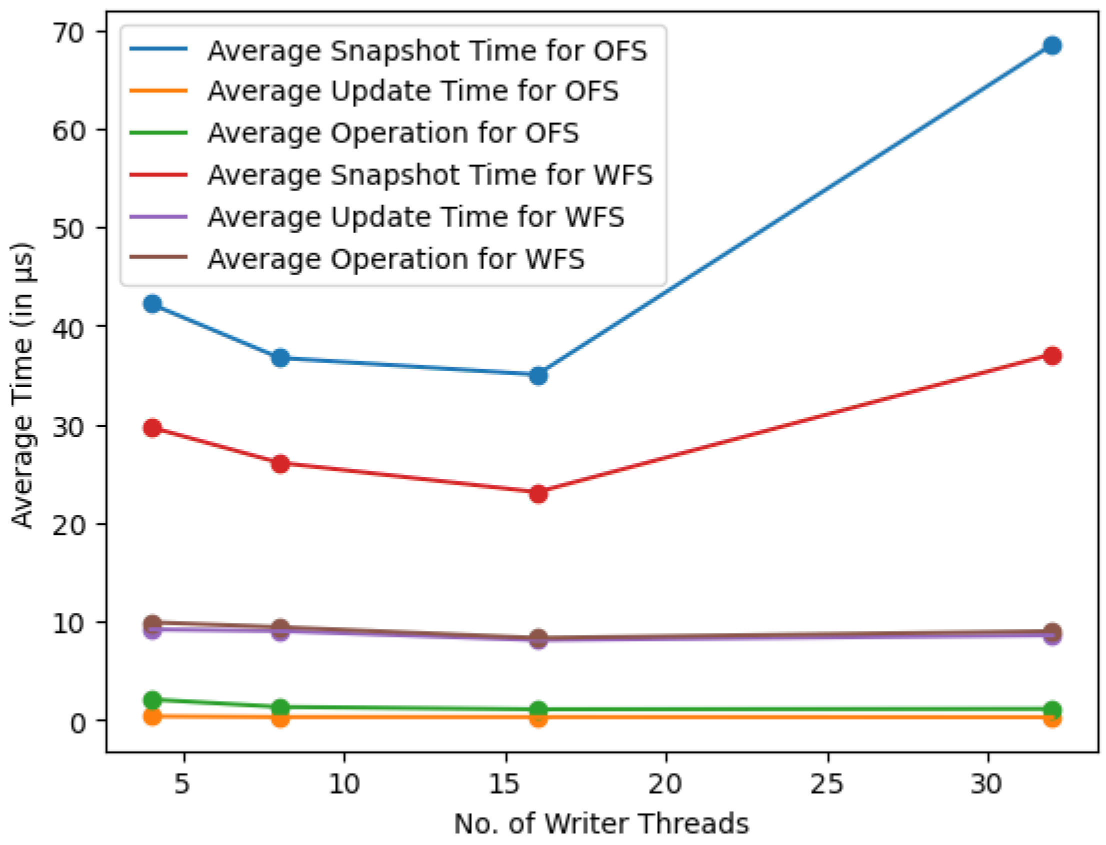
Used to simulate snapshot threads. The function performs a series of k snapshots. In each snapshot operation, it records the start time and end time of the snapshot using the gettimeofday function. It calculates the duration of the snapshot and stores it in microseconds in a vector (snTime) for tracking snapshot times. Then it retrieves the current time and generates a log message describing the snapshot operation. After completing each snapshot operation, the function sleeps for a random duration (t) determined by the exponential distribution.

Main function:

The main function reads the input from the file and creates nW writer threads and nS snapshot threads. It then waits for all the threads to finish and then computes the average time taken and worst time taken using computeStats function. It also writes the logs generated by threads to the log file.

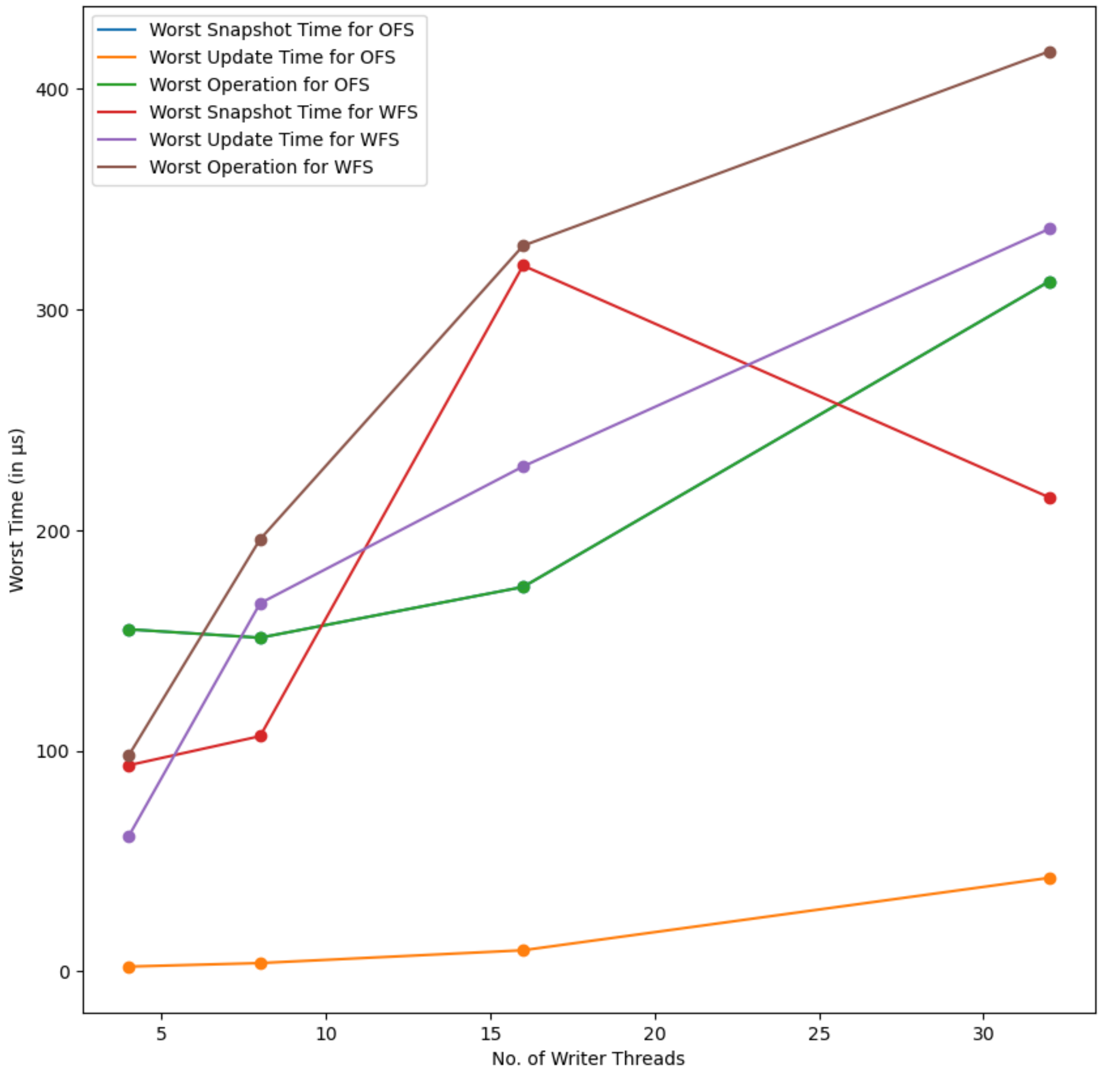
Results:

Average time vs number of writer threads (keeping nw/ns = 4, M = 40, μw = 0.5, μs = 0.5, k = 5)



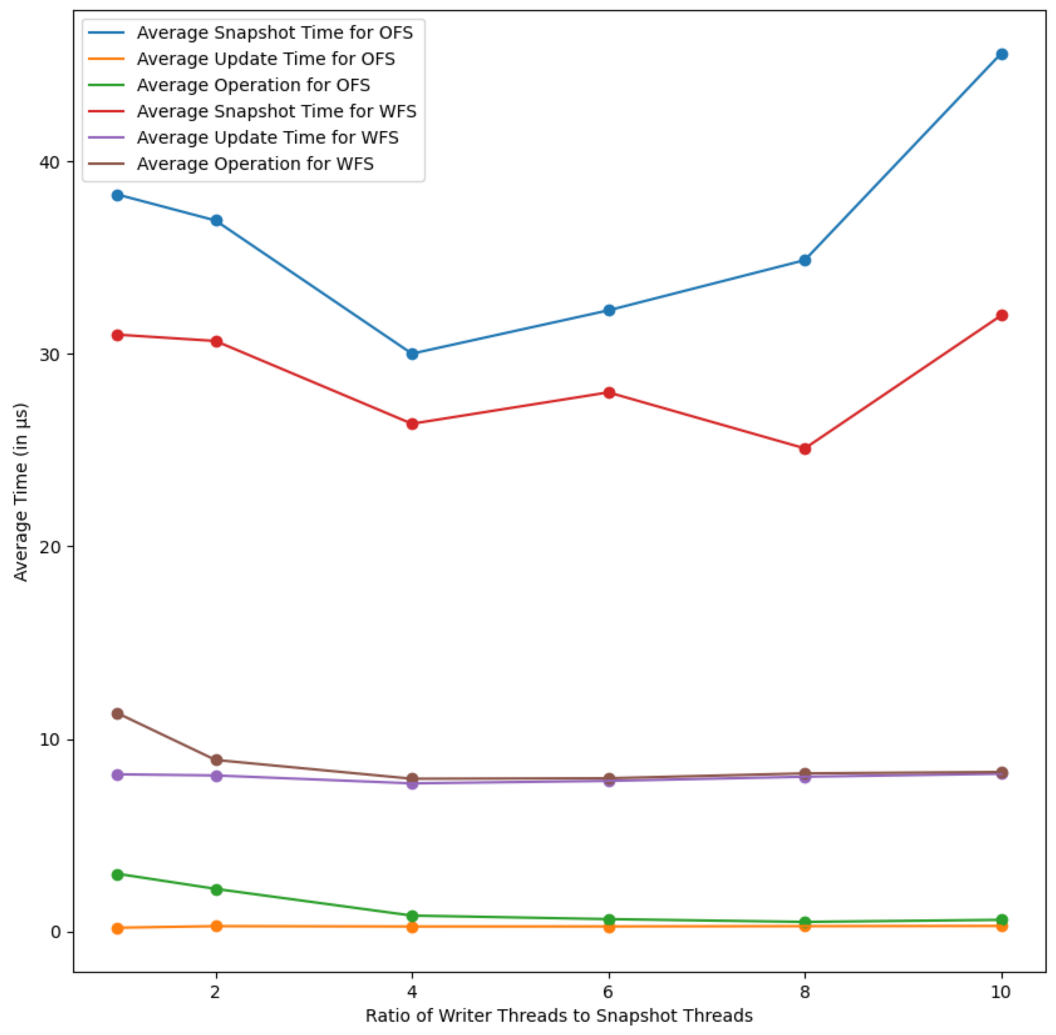
- For all the cases, average snapshot time for OFS is more than WFS as expected.
- For WFS, the average update time is more than that for OFS. This is since in WFS, each update also performs a scan call.
- Average operation time is mostly determined by the write operations since the number of write operations is much more than the number of snapshot operations.
- WFS average time for both snapshot and writes remains almost constant with increase in number of threads since the construction is wait free.
- OFS write average time remains constant since write methods are wait free but on increasing number of threads, the average snapshot time increases since more threads interfere with scan calls.

Worst time vs number of writer threads (keeping nw/ns = 4, M = 40, µw = 0.5, µs = 0.5, k = 5)



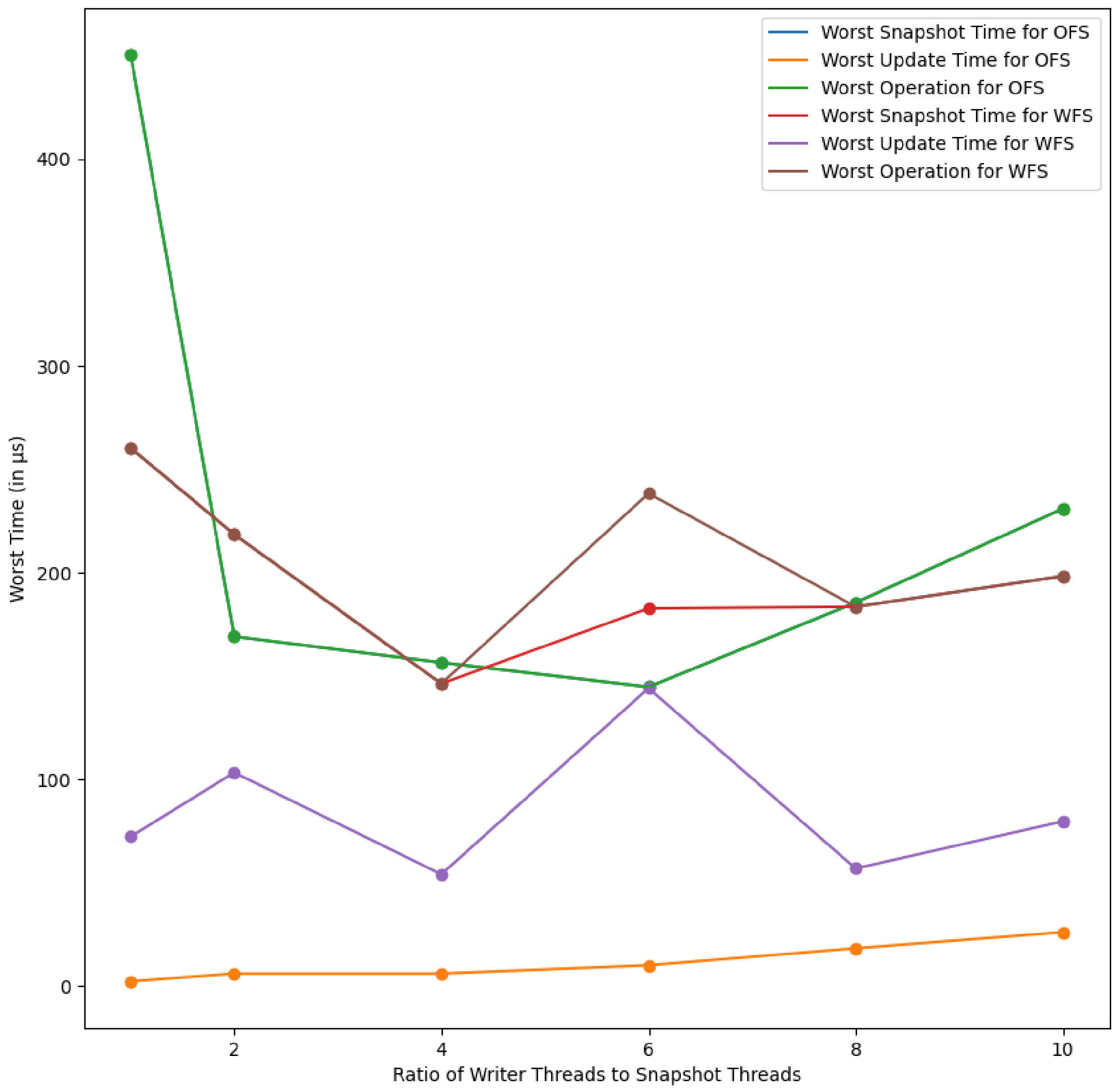
- For OFS, there is an overlap between blue (worst snapshot time) and green (worst operation time) since in all the cases, snapshots are taking much more time than writes.
- For WFS, in some cases, writes take more time than snapshots in the worst case.
- Worst time increases with increasing number of writer threads since there are more threads that interfere with scan calls.

Average time vs ratio of writer threads to snapshot threads (keeping ns = 4, M = 40, μw = 0.5, μs = 0.5, k = 5)



- For all the cases, average snapshot time for OFS is more than WFS as expected.
- For WFS, the average update time is more than that for OFS. This is since in WFS, each update also performs a scan call.
- Average operation time is mostly determined by the write operations since the number of write operations is much more than the number of snapshot operations.
- WFS average time for both snapshot and writes remains almost constant with increase in number of threads since the construction is wait free.
- OFS write average time remains constant since write methods are wait free but on increasing number of threads, the average snapshot time increases since more threads interfere with scan calls.

Worst time vs ratio of writer threads to snapshot threads (keeping nw/ns = 4, M = 40, μw = 0.5, μs = 0.5, k = 5)



- For OFS, there is an overlap between blue (worst snapshot time) and green (worst operation time) since in all the cases, snapshots are taking much more time than writes.
- For WFS, in some cases, writes take more time than snapshots in the worst case.
- Worst case for OFS is more than WFS for most cases.