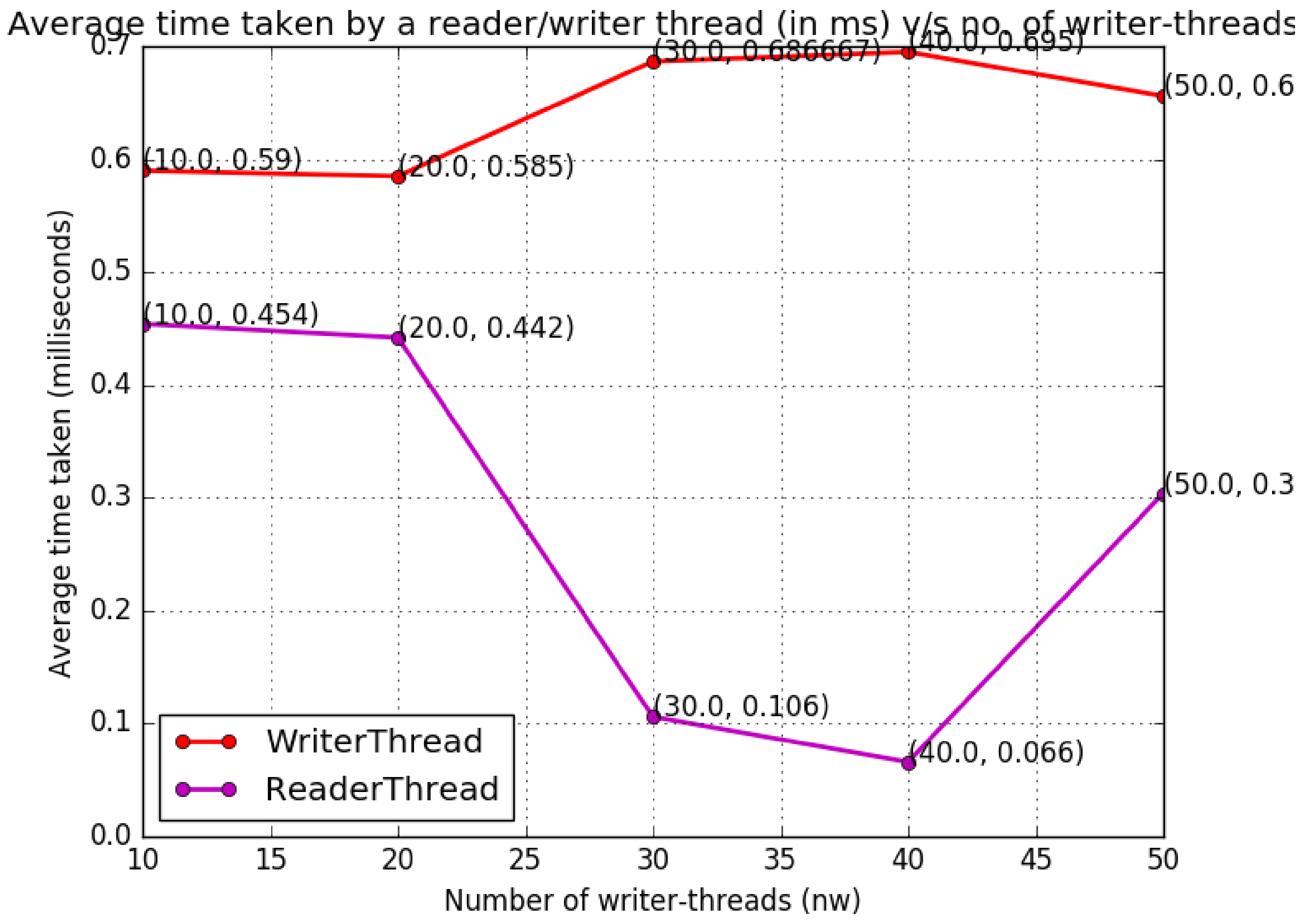
Readers & Writers Problem

Report

Aim : **To implement the solution to Readers & Writer problem, and compare the average time taken by the reader & writers threads.**

Graph representing average time taken by a reader/writer thread (in milliseconds) to complete its execution v/s number of writer threads (nw) :



Reader-Writer problem-statement : Suppose we have some reader and writer threads, where both are writing to/reading from a shared buffer concurrently. There exists a need for a constraint such that no thread is reading from/writing to the shared buffer while there is some thread writing to the buffer. However, two or more readers are allowed to access the share for reading purposes.

Solution to the above problem :

We solve this problem by ensuring that no reader is kept waiting unless a writer has already obtained permission to use the shared buffer, or we can say, no reader should wait for other readers as well to finish simply because a writer is waiting. This leads to **starvation** of writer-threads. We use two semaphores to implement the solution to reader-writer problem.

We need to satisfy the following two conditions :

- ➔ No reader will be kept waiting unless a writer has the object.
- ➔ Writing is performed as soon as possible, i.e., writers have precedence over readers.

Observations from the graph :

The above graph represents the average time taken a reader thread as well as a writer thread to finish its execution v/s the number of writer threads, which we vary from 10 to 50.

We can clearly observe that writer thread takes more time than a reader thread, because of the **starvation** that occurs with the writer threads, as well as, the reader threads can concurrently access the shared buffer, thus not waiting for each other reader threads, when the writer thread is waiting. Thus, writer thread takes more time, due to the wait time incurred because of its starvation, and hence takes more time than the reader threads.

Anomalies observed :

Since the execution of multi-threaded programs involving semaphores as above depends on the **system/CPU state**, the average time taken by each thread may not be propotional to the number of writer threads, as we can see clearly in the case of reader-threads, where the time taken drops hugely when n(w) changes from 20 to 30. Same can be observed in the case of writer-threads also, where the average time taken by a writer thread drops slightly when n(w) changes from 40 to 50.