# Report

Implementation

Of

Multi-Reader Multi-Writer

M-valued Atomic Register

**Name:** Rohit kapoor

**Roll no:** cs21mtech12011

## Program Description

In order to construct m-valued MRMW atomic register from SRRW regular register, it is required to construct other registers in between these include atomic SRSW and atomi MRSW. Further a class of time stamps needs to be created so that the classes of these registers can inherit thier methods.

For the main method a no of threads are created which perform either read or write operations on the shared variables as they enter in the function, since the variable is atomic the value associated with it remains the latest value that has been written onto the variable. For each such call we evaluate the amount of time taken in order to perform the read or write operations in nanosecond.

We evaluate the average amount of time taken for:-

- Read operations
- write operations
- Both read and write operations

These values are obtained for both the in-built atomic register and the one we created in this project.

The stats are obtained by varying the number of threads from 10 to 50 while keeping the value of k (i.e. the no of operations a thread is allowed to perform) to 100, the total no of operations performed will be equal to N*k (product of the no of threads and the no of operations each thread performs).

For avoiding the outliers, the value at each point is obtained by evaluating the avg. time at each point 5 times and then taking the average of them

## Stats

The following table depicts the variation of the avg. time taken to perform the read and write operations as the no of threads are varied from 10 to 50 (keeping k fixed at 100), lambda1=20. P=0.5).

| No of Threads | Avg. Read (micro sec) | Avg. Write (micro sec) | Avg. Read + write (milli sec) |
|---|---|---|---|
| 10 | 33.664 | 41.119 | 375.192 |
| 20 | 54.958 | 58.486 | 566.241 |
| 30 | 74.637 | 79.431 | 769.775 |
| 40 | 113.528 | 109.972 | 1115.46 |
| 50 | 146.434 | 142.454 | 1443.536 |

Table 1.1: The stats for Inbuilt atomic Register provided by C++

| No of Threads | Avg. Read (micro sec) | Avg. Write (micro sec) | Avg. Read + write (milli sec) |
|---|---|---|---|
| 10 | 17.317 | 16.06 | 166.612 |
| 20 | 38.512 | 40.293 | 393.721 |
| 30 | 47.596 | 47.443 | 474.894 |
| 40 | 75.380 | 74.512 | 749.227 |
| 50 | 97.270 | 102.047 | 995.808 |

Table 1.2: The stats for the atomicMRMW implemented in the project.

## Graph

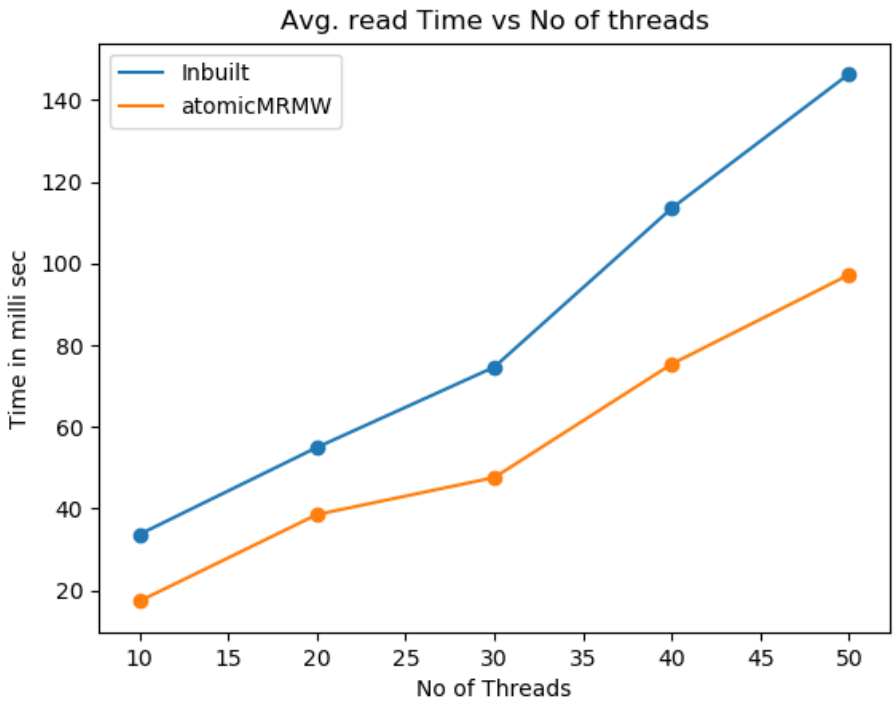


Fig. 1.1: Average read time vs no of threads

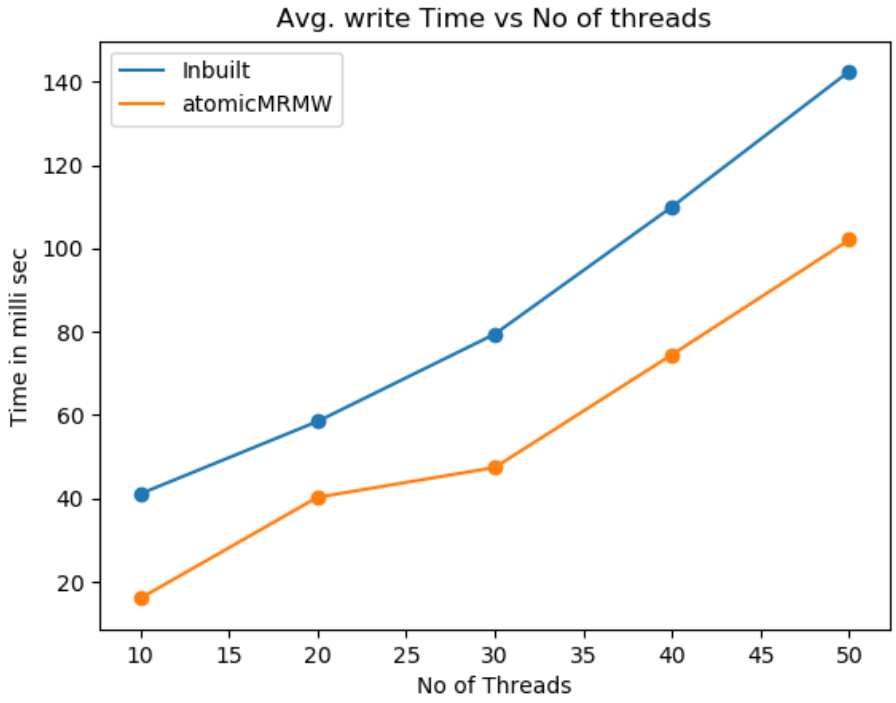


Fig. 1.2 : Average write time vs no of threads

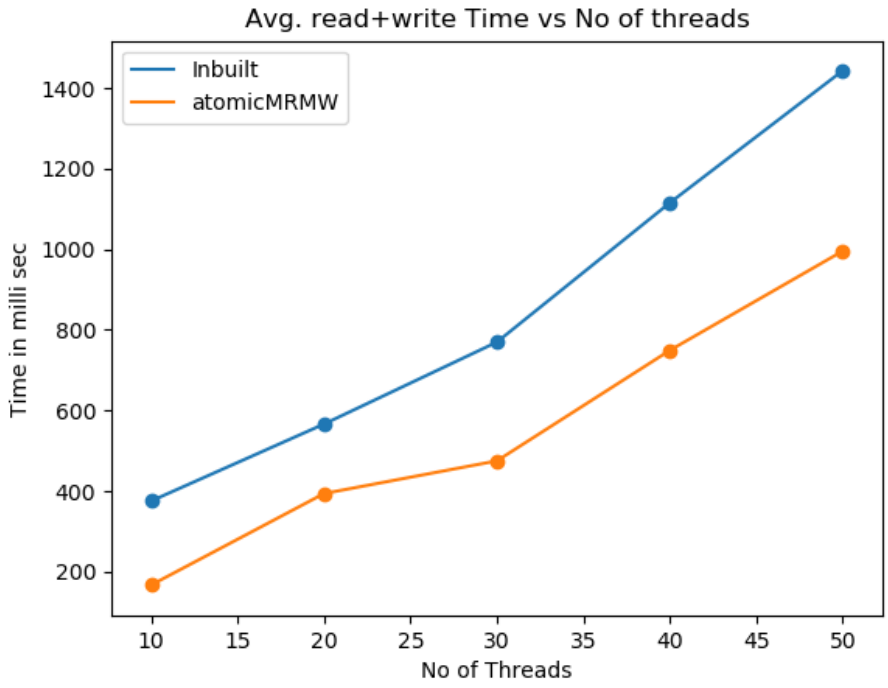


Fig. 1.3 : Variation of the average read + write time with the no of threads.

**Conclusion:**

Almost similar trend is shown in all the cases and the program created in this project is performing better, it may be the result of lesser dependencies.