Wait-Free Atomic Snapshot Implementations Comparing the Solutions of MRSW and MRMW

Design

- **MRSW**
  - WFSnapshot class was implemented given in the textbook("The Art of multiprocessor programming") which assumed AtomicMRSW registers to be available
  - AtomicMRSW class was implemented in CPP as given in the textbook
  - To satisfy the underneath properties of the variables of the class a additional class was implemented (AtomicSRSWRegister)
  - AtomicMRSW assumes AtomicSRSWRegisters are available and AtomicSRSW assumes RegularSRSW to be available
  - We assume that CPP variables are RegularSRSW
  - All of the above classes were implemented by taking reference from the textbook

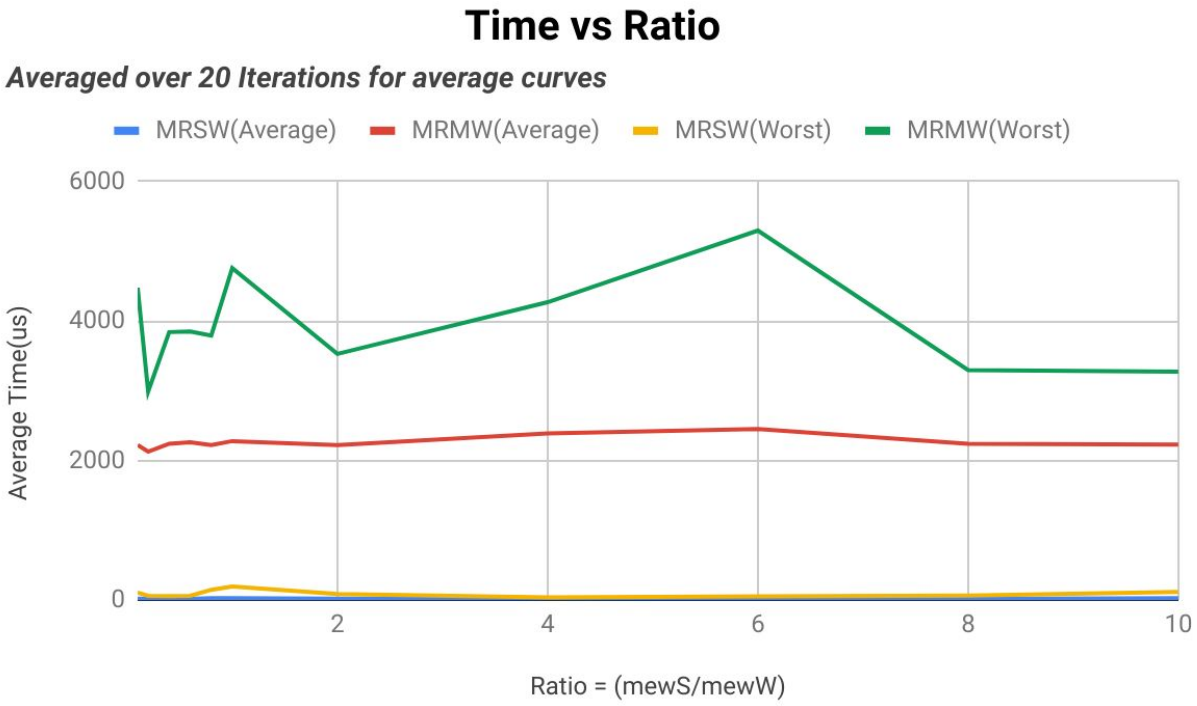
- **MRMW**
  - AtomicMRMW class was implemented instead of AtomicMRSW following the class description given in the textbook
  - WFSnapshot was implemented for AtomicMRMW registers
  - The design of the program was implemented as given in the paper, A Simple Snapshot Algorithm for Multicore Systems by D. Imbs and M. Raynal

Note1: The snapshot algorithms implemented are wait-free
Note2: I was not able to use inbuilt atomic variables directly while constructing the WFSnapshot class, this is discussed more in the Difficulties faced section

Graphs and Analysis

n = 10 (Number of threads)
k = 5 (Number of time to take snapshot)
Ratio = mewS/mewW = [10, 8, 6 .. 2, 1, 0.8 .. 0.2, 0.1] (ratio of average delay)

★ Each test was run 20 times and the average value was calculated to plot the graphs



- As can be seen in the above graph, the time taken for the snapshot algorithm is more with AtomicMRMW registers even though the testing was done assuming only MRSW access with AtomicMRMW registers too
- This large time for AtomicMRMW can be explained by the fact that more work (computationally) is done to meet the requirements of AtomicMRMW than AtomicMRSW, as can be seen from the definitions of the class

Ratio	MRSW(Average)	MRMW(Average)	MRSW(Worst)	MRMW(Worst)
0.1	18.1	2219.15	105	4475
0.2	16.80952381	2125.285714	58	2985
0.4	16.71428571	2238.095238	54	3839
0.6	15.23809524	2258.190476	58	3848
0.8	22.76190476	2217.714286	145	3788
1	25.76190476	2275.952381	192	4757
2	15.66666667	2218.714286	85	3526
4	12.47619048	2387.238095	35	4267
6	16	2449.904762	52	5294
8	14.76190476	2237.619048	61	3295
10	24.95238095	2225.190476	113	3272

Note1: The above table contains time in microseconds

Difficulties faced:

- atomic variables cannot be used as variables and initialized while implementing a class in cpp, because of this I needed to implement all the underlying classed to satisfy the property of the registers assumed to be available in the base class(WFSnapshot)
- The following error was encountered:

  error: use of deleted function `std::atomic<_Tp>::atomic() [with _Tp = **ClassName**]'

- It is a known bug identified in gnu, can be seen here: Bug

Correctness of the Snapshot Algorithm:

We can confirm the correctness of the program by observing that each snapshot present in the log files was a state the system was present at some point of time

MRMW:

```
Snapshot Thr's snapshot: t0 - 0 t1 - 0 t2 - 0 t3 - 0 t4 - 0 t5 - 0 t6 - 0 t7 - 0 t8 - 0 t9 - 0 which finished at 18:42:11
Thr2's write of 846930886 at 18:42:11
Thr3's write of 1714636915 at 18:42:11
Thr1's write of 1804289383 at 18:42:11
Thr6's write of 1649760492 at 18:42:11
Thr9's write of 1189641421 at 18:42:11
Thr7's write of 596516649 at 18:42:11
Thr0's write of 1681692777 at 18:42:11
Thr4's write of 1957747793 at 18:42:11
Thr5's write of 424238335 at 18:42:11
Thr8's write of 719885386 at 18:42:11
Snapshot Thr's snapshot: t0 - 1681692777 t1 - 1804289383 t2 - 846930886 t3 - 1714636915 t4 - 1957747793 t5 - 424238335
t6 - 1649760492 t7 - 596516649 t8 - 719885386 t9 - 1189641421 which finished at 18:42:11
Snapshot Thr's snapshot: t0 - 1681692777 t1 - 1804289383 t2 - 846930886 t3 - 1714636915 t4 - 1957747793 t5 - 424238335
t6 - 1649760492 t7 - 596516649 t8 - 719885386 t9 - 1189641421 which finished at 18:42:11
```

MRSW:

```
Thr2's write of 846930886 at 20:15:5
Snapshot Thr's snapshot: t0 - 0 t1 - 0 t2 - 846930886 t3 - 0 t4 - 0 t5 - 0 t6 - 0 t7 - 0 t8 - 0 t9 - 0 which finished at 20:15:5
Thr0's write of 1681692777 at 20:15:5
Snapshot Thr's snapshot: t0 - 1681692777 t1 - 0 t2 - 846930886 t3 - 0 t4 - 0 t5 - 0 t6 - 0 t7 - 0 t8 - 0 t9 - 0 which finished at
20:15:5
Snapshot Thr's snapshot: t0 - 1681692777 t1 - 0 t2 - 846930886 t3 - 0 t4 - 0 t5 - 0 t6 - 0 t7 - 0 t8 - 0 t9 - 0 which finished at
20:15:5
Snapshot Thr's snapshot: t0 - 1681692777 t1 - 0 t2 - 846930886 t3 - 0 t4 - 0 t5 - 0 t6 - 0 t7 - 0 t8 - 0 t9 - 0 which finished at
20:15:5
Thr1's write of 1804289383 at 20:15:5
Snapshot Thr's snapshot: t0 - 1681692777 t1 - 1804289383 t2 - 846930886 t3 - 0 t4 - 0 t5 - 0 t6 - 0 t7 - 0 t8 - 0 t9 - 0 which
finished at 20:15:5
```

The above snippets show that indeed the snapshots taken by the algorithm existed in the system state before it

Note: The above snippets were taken from the logs obtained from the execution of the programs