Sri Hari Malla - CS19BTECH11039
Prof : Dr. Sathya Peri
Parallel and Concurrent Programming
Indian Institute of Technology Hyderabad

PROGRAMMING ASSIGNMENT - IV

Sri Hari Malla - CS19BTECH11039

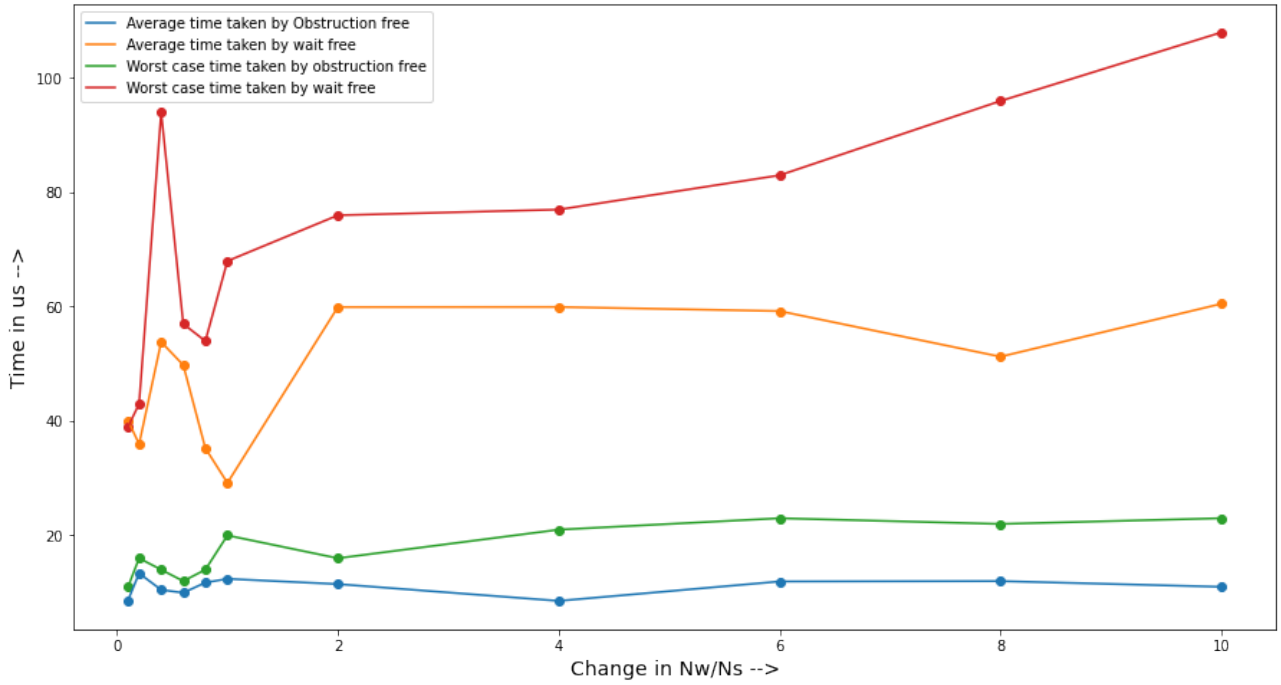
April 2, 2022

Base:

The application was designed exactly as described in the programming assignment problem statement and used text book for the reference. I implemented Obstruction free and Wait free registers from the `SnapShot` interface, `StampedSnap` and `StampedValue` using `C++`.

Graph & Observations:

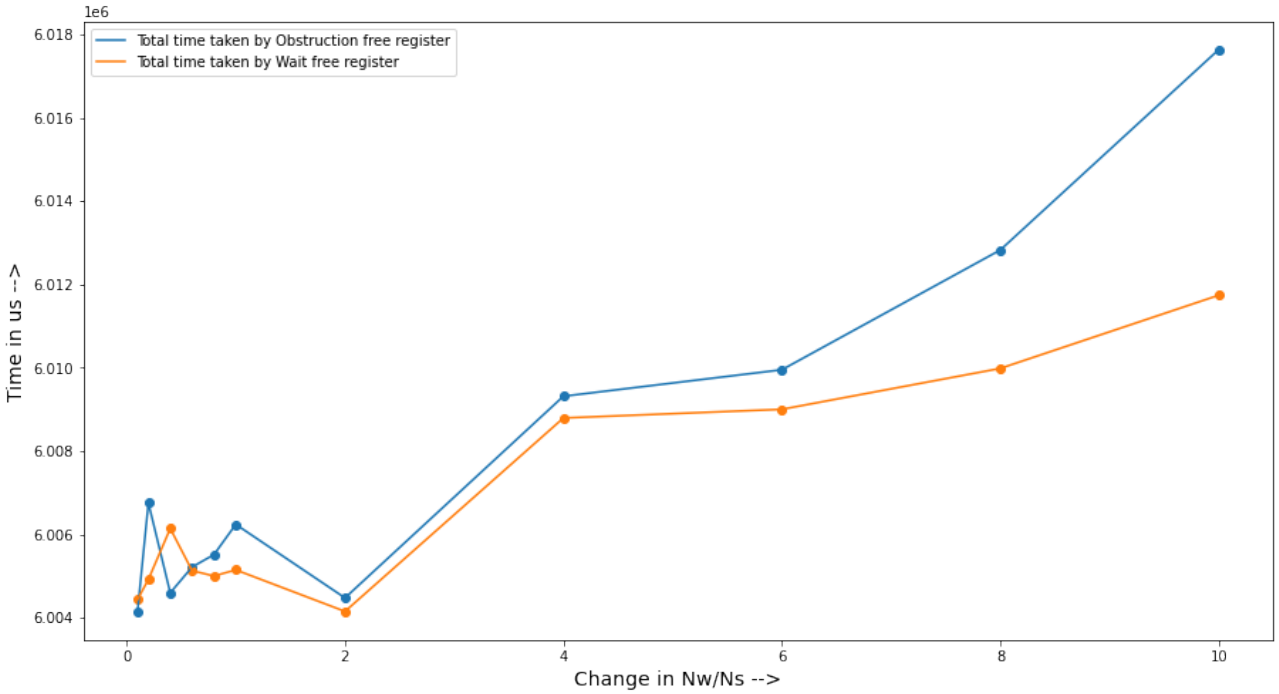
Graph: Variation in Nw/Ns vs Variation in time(for both registers, both average and worst case)



Every observation is the average of 5 recorded observations to remove any discrepancies in the measuring of time. The Nw/Ns is varied from 0.1 to 10 while keeping repetition count as constant and set to 5 while the uw and us are set to 0.5 respectively.

Observations:

- The graphs were plotted by recording times for average and worst case and for for both the registers.
- One can see in the graph that the obstruction free register is taking lesser time for an average snap operation when compared to wait free register.
- This is the observed anomaly, where the wait free register should be faster than obstruction free register.
- The device I used is included with a lot of uncertainty because of multiple processes running in parallel.
- The complexity is some what higher for one scan operation in wait free than in the obstruction free register and this could possibly the reason for the anomaly detected.



- This is the plot of total time taken for one execution set of Obstruction free register and wait free register.
- It is clear from this graph that wait free register is faster than the obstruction free register.
- Small anomalies are observed at very less Nw/Ns.
- The graph is consistent with the observations reported in problem statement.

Validating the results:

```
Obstruction free snapshot register logs :
Writer log :
Thread t0 write of 183 on location 6 at 17:55:13.492559
Thread t0 write of 377 on location 15 at 17:55:13.492901
Thread t0 write of 193 on location 15 at 17:55:13.493105
Thread t0 write of 186 on location 12 at 17:55:13.493360
Thread t0 write of 249 on location 1 at 17:55:13.493581
Thread t0 write of 362 on location 7 at 17:55:13.493811
Thread t0 write of 290 on location 19 at 17:55:13.494030
Snapper log :
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-183 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-0 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.492640
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-183 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-377 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.492912
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-183 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-377 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.493107
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-183 L7-0 L8-0 L9-0 L10-0 L11-0 L12-186 L13-0 L14-0 L15-193 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.493530
thread Id - 0
L0-0 L1-249 L2-0 L3-0 L4-0 L5-0 L6-183 L7-0 L8-0 L9-0 L10-0 L11-0 L12-186 L13-0 L14-0 L15-193 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.493783
```

- One can see the screenshot of Obstruction free register log attached above.
- It is clear from the graph that the snapper log(log of snap shots) is consistent, i.e linearizable.
- For every time stamp at the snapper log, the value at the previously reflected timestamps at writerlog is observed perfectly.
- The time stamp taken in `milliseconds` has a lot of uncertainity, so I used `microseconds` and the results observed are perfect.

```
Wait free snapshot register logs :
Writer log :
Thread t0 write of 163 on location 6 at 17:55:13.517635
Thread t0 write of 140 on location 6 at 17:55:13.518081
Thread t0 write of 372 on location 16 at 17:55:13.518272
Thread t0 write of 11 on location 8 at 17:55:13.518621
Thread t0 write of 167 on location 9 at 17:55:13.519000
Thread t0 write of 182 on location 10 at 17:55:13.519307
Snapper log :
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-163 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-0 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.517710
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-163 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-0 L16-0 L17-0 L18-0 L19-0
recorded at 17:55:13.517982
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-140 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-0 L16-372 L17-0 L18-0 L19-0
recorded at 17:55:13.518299
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-140 L7-0 L8-0 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-0 L16-372 L17-0 L18-0 L19-0
recorded at 17:55:13.518549
thread Id - 0
L0-0 L1-0 L2-0 L3-0 L4-0 L5-0 L6-140 L7-0 L8-11 L9-0 L10-0 L11-0 L12-0 L13-0 L14-0 L15-0 L16-372 L17-0 L18-0 L19-0
recorded at 17:55:13.518798
```

- One can see the screenshot of Wait free register log attached above.
- It is clear from the graph that the snapper log(log of snap shots) is consistent, i.e linearizable.
- For every time stamp at the snapper log, the value at the previously reflected timestamps at writerlog is observed perfectly.
- The time stamp taken in `milliseconds` has a lot of uncertainity, so I used `microseconds` and the results observed are perfect.
- This is a sample run observed for only one writer thread and one collector thread(more threads gives very lengthy screenshots).

LaTeX generated document

THE END