**CS4532 Concurrent Programming Lab 2**







**Case 1**

n = 1,000 and m = 10,000, mMember = 0.99, mInsert = 0.005, mDelete = 0.005

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| implementation | No of Threads | | | | | |
| 1 | | 2 | | 4 | |
| Average | Std | Average | Std | Average | Std |
| Serial | 0.018292 | 0.001462 |  | | | |
| Mutex | 0.018008 | 0.001238 | 0.052919 | 0.001187 | 0.056658 | 0.001273 |
| Read-Write Lock | 0.019514 | 0.001662 | 0.011193 | 0.000335 | 0.00694 | 0.000532 |

**Case 2**

n = 1,000 and m = 10,000, mMember = 0.90, mInsert = 0.05, mDelete = 0.05

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| implementation | No of Threads | | | | | |
| 1 | | 2 | | 4 | |
| Average | Std | Average | Std | Average | Std |
| Serial | 0.024145 | 0.002024 |  | | | |
| Mutex | 0.024765 | 0.002287 | 0.061295 | 0.00527 | 0.062777 | 0.002444 |
| Read-Write Lock | 0.02236 | 0.001432 | 0.020788 | 0.001778 | 0.017507 | 0.001427 |

**Case 3**

n = 1,000 and m = 10,000, mMember = 0.50, mInsert = 0.25, mDelete = 0.25

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| implementation | No of Threads | | | | | |
| 1 | | 2 | | 4 | |
| Average | Std | Average | Std | Average | Std |
| Serial | 0.048998 | 0.003958 |  | | | |
| Mutex | 0.046495 | 0.002356 | 0.087977 | 0.002994 | 0.092543 | 0.002928 |
| Read-Write Lock | 0.053186 | 0.003649 | 0.071601 | 0.002727 | 0.072557 | 0.006455 |

**System Specifications**

Operating System – OS X Sierra 10.12.4

CPU – 2.2 GHz Intel Core i7 (4 physical cores)

Memory – 16 GB RAM

Comments:

* The execution time of the mutex increases with the thread count due to many threads are blocked at the same time to acquire the lock. The same conclusion can be applied to the Read Write Lock.
* Read write lock shows significance performance than the mutex (A read-write lock allows only one writer but you can have many readers) where mutex allow access only to a one thread.