**CS4532 Concurrent Programming - Take Home Lab 2**

1. Design a solution such that you can generate m Member , m Insert , and m Delete operations of each type using given number of threads. Briefly explain your design.

**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(s)** | **Std** | **Average(s)** | **Std** | **Average(s)** | **Std** |
| **Serial** | 0.018910 | 0.002571 |  | | | |
| **One mutex for entire list** | 0.020877 | 0.001968 | 0.033928 | 0.005010 | 0.039798 | 0.005708 |
| **Read – Write lock** | 0.001581 | 0.003577 | 0.003945 | 0.001522 | 0.003391 | 0.001520 |

**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(s)** | **Std** | **Average(s)** | **Std** | **Average(s)** | **Std** |
| **Serial** | 0.025627 | 0.002141 |  | | | |
| **One mutex for entire list** | 0.027628 | 0.000689 | 0.042103 | 0.002905 | 0.050159 | 0.001976 |
| **Read – Write lock** | 0.005886 | 0.007223 | 0.007544 | 0.003673 | 0.007405 | 0.004105 |

**Case 3:**

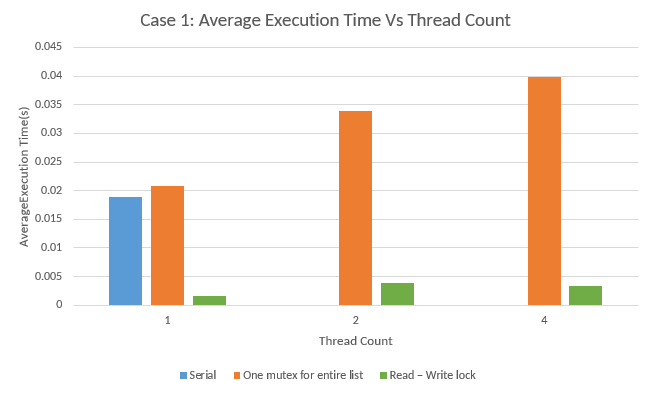
*n* = 1,000 and *m* = 10,000, *mMember* = 0.50, *mInsert* = 0.25, *mDelete* = 0.25

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Implementation** | **No of threads** | | | | | |
| **1** | | **2** | | **4** | |
| **Average(s)** | **Std** | **Average(s)** | **Std** | **Average(s)** | **Std** |
| **Serial** | 0.058258 | 0.000839 |  | | | |
| **One mutex for entire list** | 0.062113 | 0.002562 | 0.083247 | 0.002577 | 0.084817 | 0.002888 |
| **Read – Write lock** | 0.031889 | 0.004178 | 0.046068 | 0.006501 | 0.055693 | 0.005667 |

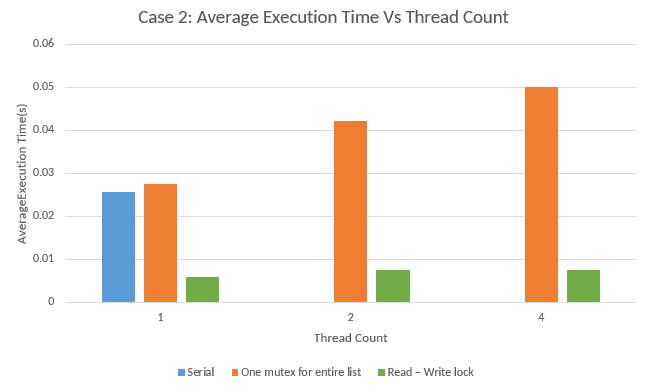
Specifications of the machine:

* Operating System – Ubuntu 15.10 64 bit
* CPU – Intel Core i7 4700MQ CPU@2.40GHzx8
* Memory – 8GB RAM

**Case 1:**



**Case 2:**



**Case 3:**



Comments:

* It can be concluded that the serial program has better performance over the program with a mutex.
* Also the read-write lock has better performance over both the mutex and the serial program. This is because mutex uses only a single lock for the whole linked list. This allows only one read operation to execute at a time where as the read write lock allows multiple reads to execute simultaneously. Threads have to waste a significant amount of time while being blocked to acquire the lock.
* Furthermore, it is seen that the execution time of the program with read-write lock increases as the fractions of insert, delete operations increases. The reason for this outcome could be explained as follows. Although read-write locks allows multiple reads simultaneously, it allows only one write operation at a time. As the fractions of insert, delete operations increase, the number of write operations needed to be performed increase alongside. Hence the execution time increases.
* The execution time of the mutex increases with the thread count since many threads are blocked at the same time to acquire the lock. The same conclusion can be derived with the read-write lock as well. This is also due to the reason that threads are blocked at the same time while waiting to acquire the lock.
* Another observation is that the execution time of the mutex increases as the fractions of insertions and deletions increase. This can be explained since write operations take more time than the read operations.