Spin Locks

Karan Agrawal : 2023MCS2479

February 2024

1 Algorithm Explanation

1. **Classes**: Classes to implement various locks:-
   - **Lock**: A simple template class with only two function declarations *lock()* and *unlock()*.
   - **TASLock (Test-and-Set Lock):**
     - TASLock is a spinlock that uses the atomic test-and-set operation to acquire the lock.
     - It continuously spins in a loop, attempting to set a flag to acquire the lock, making it inefficient for highly contended locks.
   - **TTASLock (Test-and-Test-and-Set Lock):**
     - TTASLock is an improvement over TASLock that reduces bus contention by first checking if the lock is available before attempting to acquire it with test-and-set.
     - It helps alleviate the overhead of continuously attempting to set the lock flag by spinning only when necessary.
   - **ALock (Anderson Lock):**
     - ALock is a scalable, ticket-based spinlock designed for highly contended locks.
     - It uses a queue-based mechanism where each thread obtains a ticket and spins on its own slot, reducing contention on the lock.
   - **CLHLock (Craig, Landin, and Hagersten Lock):**
     - CLHLock is a queue-based lock that avoids spinning by allowing threads to yield their turn to acquire the lock, reducing CPU wastage.
     - It provides efficient mutual exclusion and scalability, making it suitable for use in various concurrent applications.
   - **MCSLock (Mellor-Crummey and Scott Lock):**
     - MCSLock is a queue-based lock similar to CLHLock but offers better cache performance due to its explicit queue structure.

- It ensures fairness and scalability by maintaining a separate queue node for each thread, minimizing contention and providing efficient mutual exclusion.

2. **Main Function**

   - Firstly, iterating a loop for each lock
   - Then iterating nested loop for a number of threads will run(N=20 to N=100).
   - Storing time in milliseconds for corresponding lock to number of threads.
   - then draw the graph for each lock with X-axis as number of threads and Y-axis as time.

2 Results: Graphs

- **Lock Performance Summary**

  The completion times for TASLock, TTASLock, ALock, CLHLock, and MCSLock were measured with 20-100 threads to assess scalability and efficiency.

  Key Findings:

- **Average Times:**
  - CLHLock and MCSLock had shortest completion times, showing efficient scalability.
  - TASLock and TTASLock exhibited higher completion times due to spinning-based mechanisms.
  - ALock's performance varied, generally better than TASLock and TTASLock.
- **Scalability:**
  - CLHLock and MCSLock showed consistent performance with increasing threads.
  - TASLock and TTASLock's performance deteriorated with higher thread counts.
  - ALock's scalability varied across thread counts.
- **Bottlenecks:**
  - TASLock and TTASLock had longer completion times, indicating potential inefficiencies in spinning-based locks.
  - CLHLock and MCSLock consistently showed efficient performance.

  Conclusion:

  CLHLock and MCSLock outperformed TASLock, TTASLock, and ALock in terms of efficiency and scalability.

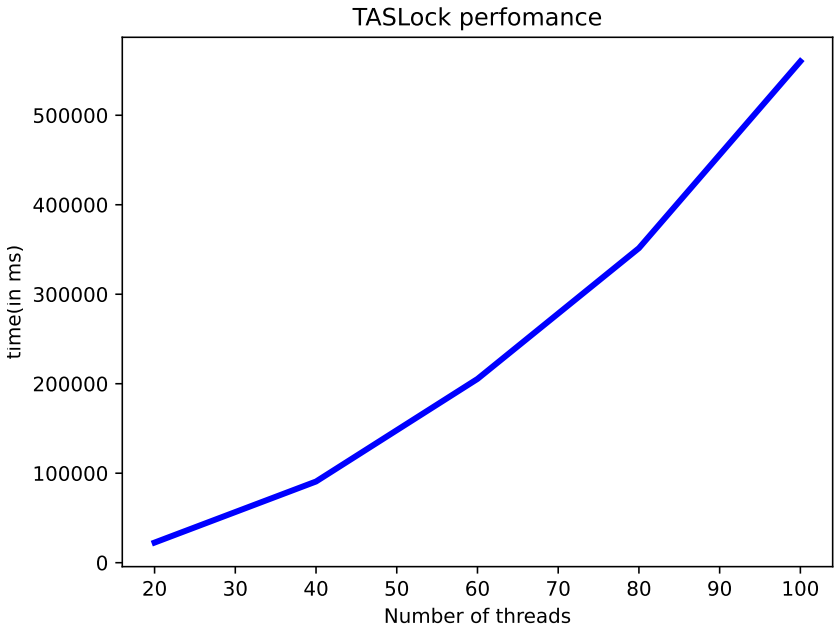

Figure 1: TASLock

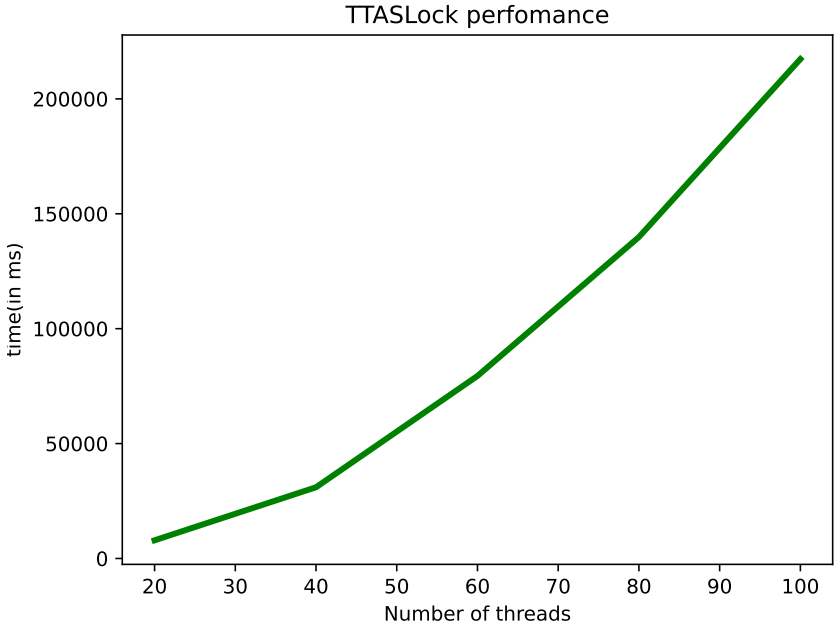

Figure 2: TTASLock

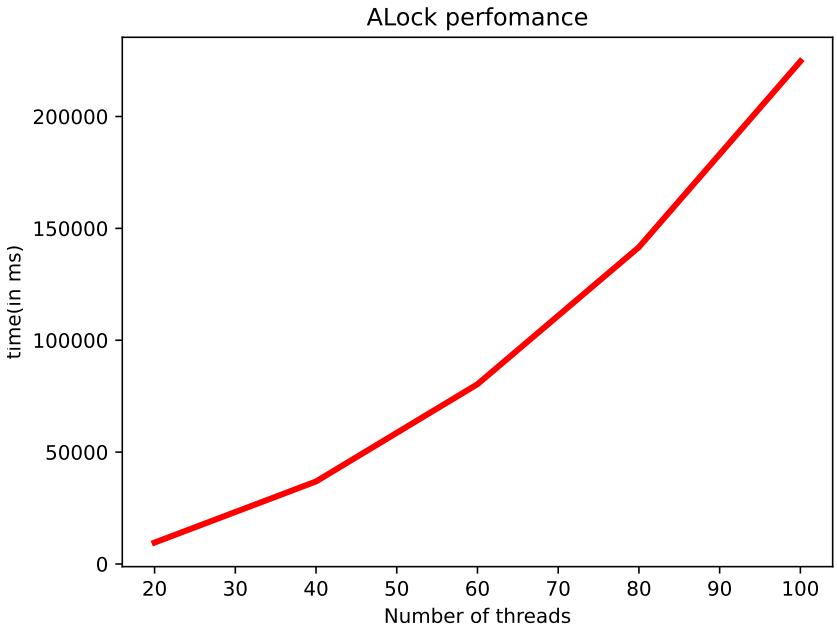

Figure 3: ALock

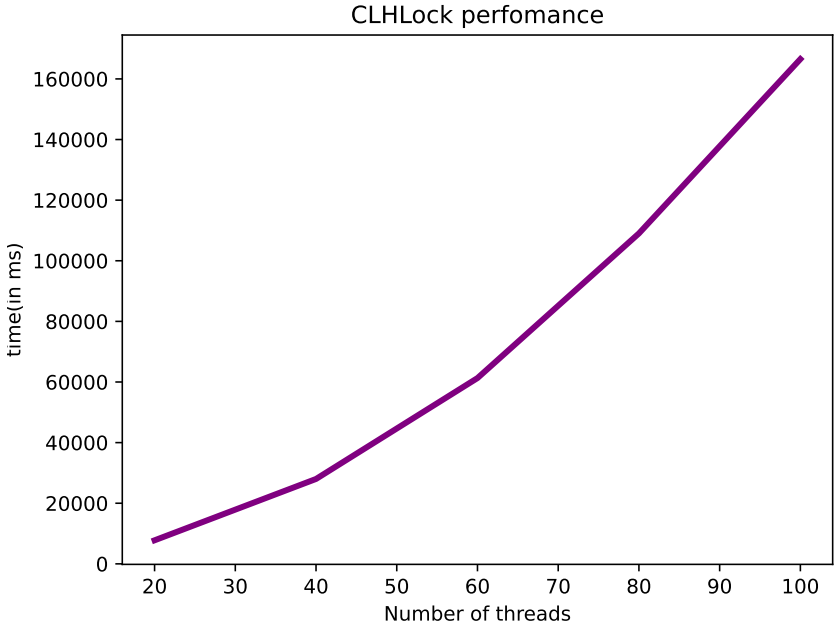

Figure 4: CLHLock

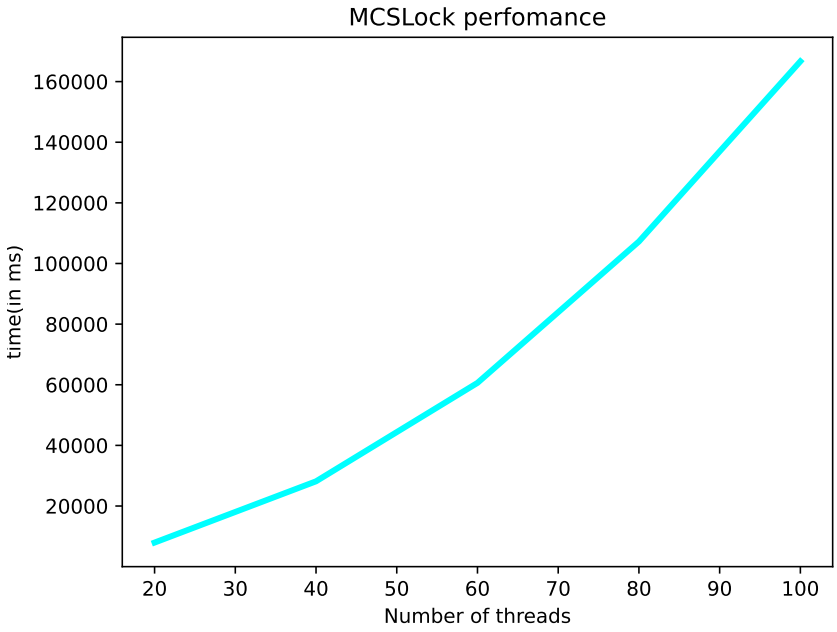

Figure 5: MCSLock

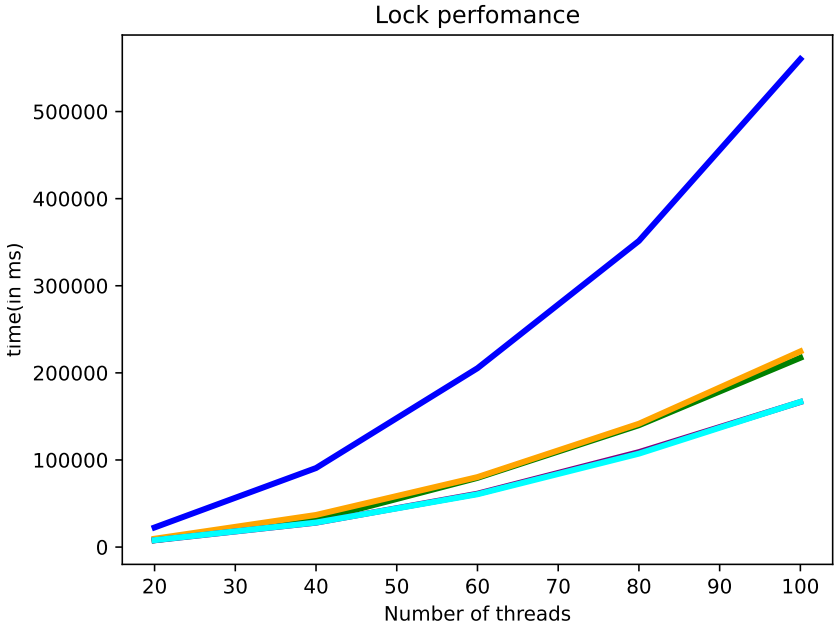

Figure 6: Locks