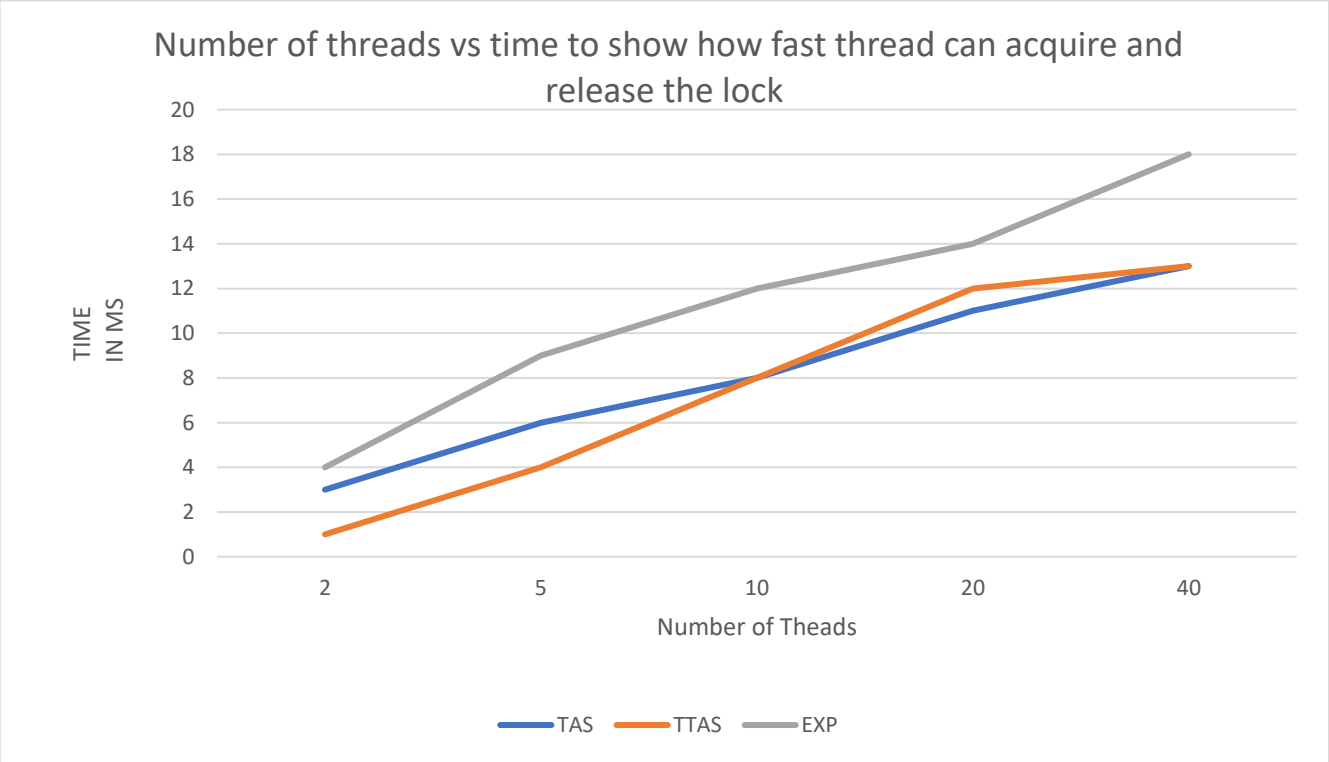
226 PRACTICAL 5

BAHIYA HOOSEN - U22598546

INTRODUCTION

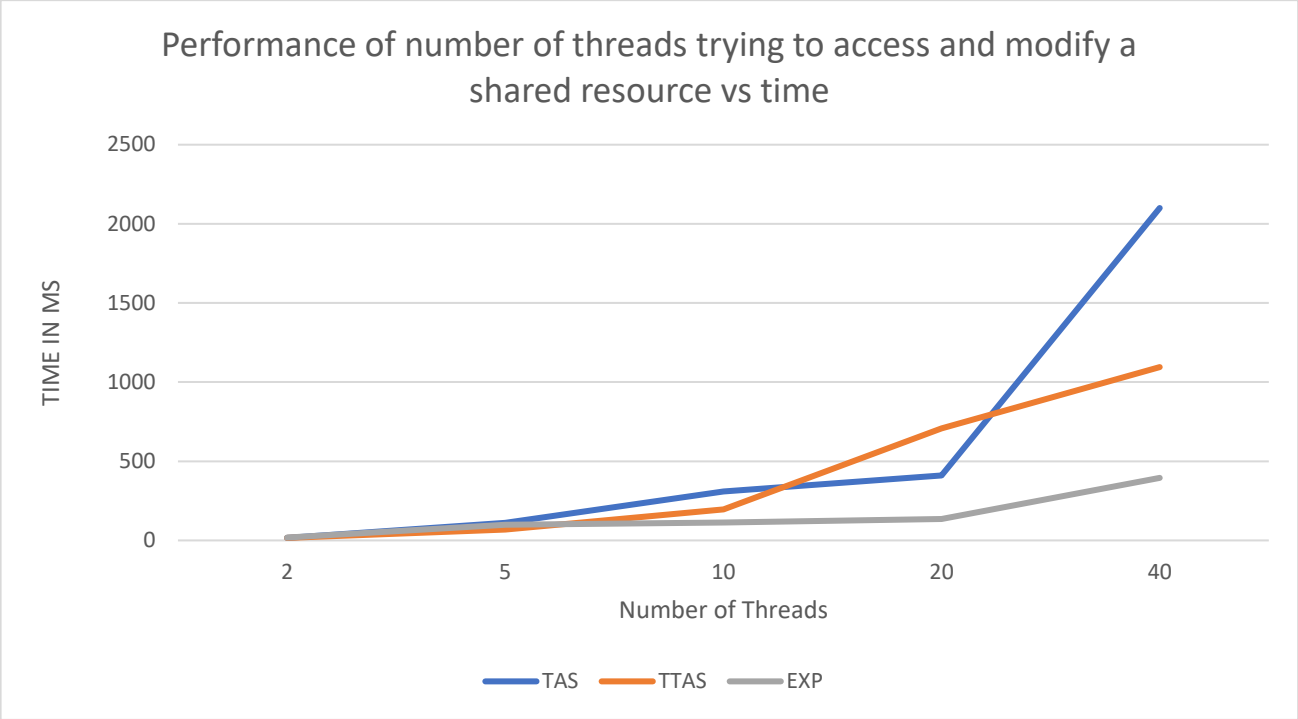
Once upon a time in the kingdom of Multithreadia, three brave locks—TAS, TTAS, and Exponential Backoff—were summoned to bring peace to the land of Chaos, where threads constantly battled for control. Each lock had its own quirky strategy: TAS was the impulsive one, always charging in without thinking; TTAS was the cautious type, peeking before making a move; and Exponential Backoff was the chill one, patiently waiting longer each time things got busy. The mission? To see who could handle the chaos best in various thread duels. Spoiler: it got messy, but hilariously so.

TEST 1: MULTIPLE THREADS ACCESSING A SHARED QUEUE




- TAS : best for small number of threads with low contention because threads can frequently acquire and release the lock without major delays. Performance degrades significantly with more threads due to high contention and wasted CPU resources on busy-waiting.
- TTAS : TTAS performs better than TAS because it avoids unnecessary busy-waiting by checking if the lock is available before attempting to acquire it. he overhead of constantly checking the lock state becomes significant as the thread count increases, leading to diminishing returns compared to TAS with a large number of threads.
- EXP : threads are unnecessarily delaying their retries, even though contention is minimal. The backoff delay proves useful, especially with higher numbers of threads. The performance scales better compared to TAS and TTAS because the backoff reduces the number of active threads competing for the lock at any given moment. However, the waiting periods also mean that performance does not scale perfectly and plateaus at higher thread counts (e.g., 40 threads).

TEST 2 : INCREASED CONTENTION WHILE MODIFYING SHARED COUNTER




- TAS : uses simple spinlock mechanism where threads constantly check and set the lock, the time taken grows dramatically especially at 10 threads due to high contention. Many threads spend most of their time spinning, waiting for the lock to be released.
- TTAS : Since TTAS first tests if the lock is available before attempting to acquire it, fewer threads spin uselessly compared to TAS. This testing reduces the number of unnecessary context switches and busy waiting, leading to more efficient lock management under moderate contention.
- EXP : When a thread fails to acquire the lock, it waits for an exponentially increasing amount of time before retrying. This reduces the frequency of lock acquisition attempts, lowering contention. The time taken to complete the task grows more gradually than with TAS or TTAS.

REFERENCES


- "The Art of Multiprocessor Programming" by Maurice Herlihy and Nir Shavit


*Roses are Red,*

*Violets are Blue*

*Unexpected ' } ' on line 32.*