PROGRAMMING ASSIGNMENT 3

Aditya Pulapalli.

PID – adityasrimannarp@vt.edu

Rohith Malkuchi.

PID – mrohi947@vt.edu

Contents

Steps to Execute 3

Deque .... **Error! Bookmark not defined.**

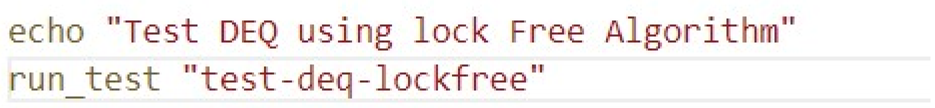
Lock Based Deque: 3

Lock-free Deque .... **Error! Bookmark not defined.**

.... **Error! Bookmark not defined.**

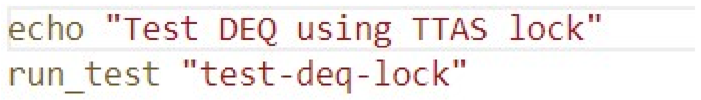
Steps to Execute

1. Run the make file to compile the executables.
Command - make
2. Run the test-fifo.sh file by using the command ./test-fifo.sh
Command - ./run-test-fifo.sh
3. Observe the results in the terminal
Note: It is recommended to run lock and lockfree independently as the output doesn't fit in the command line at once.
To run lockfree:



```
echo "Test DEQ using lock Free Algorithm"
run_test "test-deq-lockfree"
```

To run lock:



```
echo "Test DEQ using TTAS lock"
run_test "test-deq-lock"
```

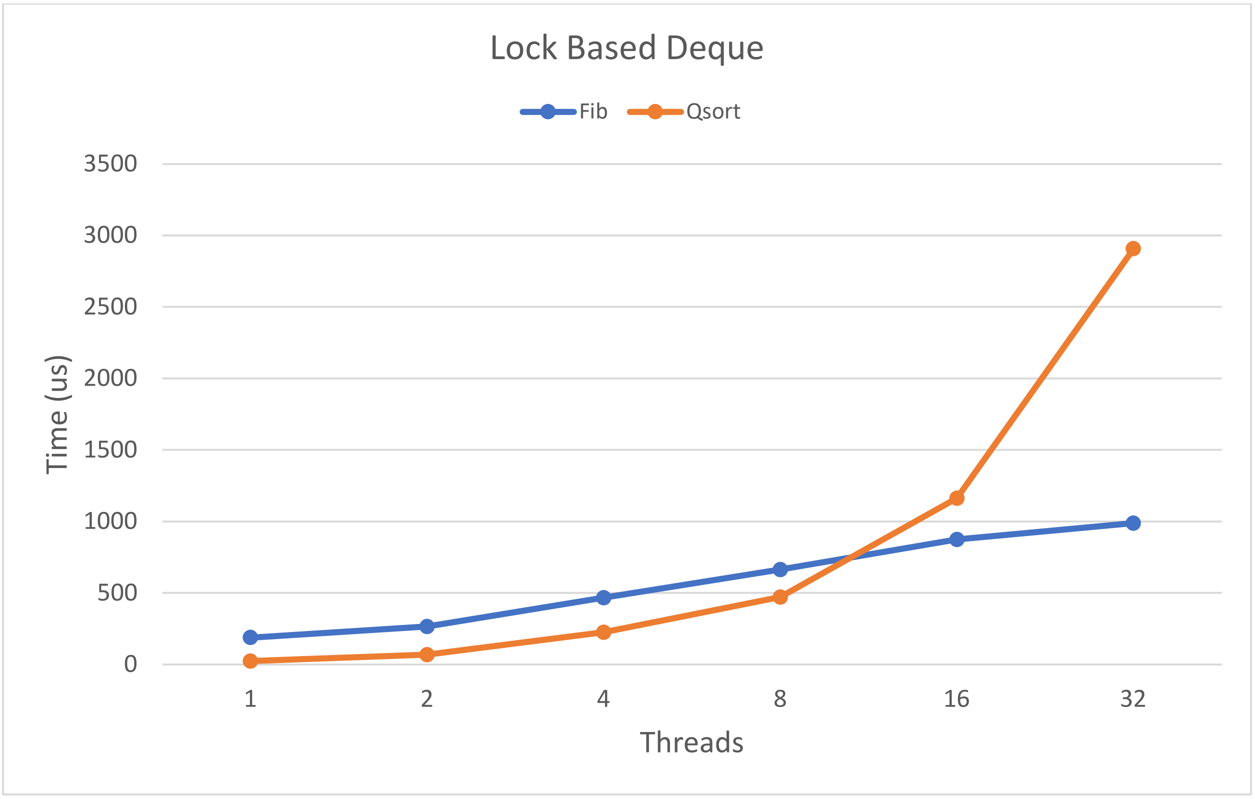
Deque

Lock Based Deque:

Lock Used: TTAS Backoff
Implementing a Test-and-Test-and-Set (TTAS) backoff lock in a deque (double-ended queue) presents several advantages, particularly in contexts where efficient, concurrent operations are crucial. Each characteristic of the TTAS lock aligns well with the operational needs of a deque, as detailed below:
1. **Reduced Bus Traffic Compared to XCHG Locks:**
   - In deques, operations like insertion and deletion at both ends are frequent. TTAS locks, by minimizing atomic operations, significantly reduce bus traffic compared to XCHG locks. This reduction in bus traffic is crucial in enhancing the efficiency of these operations, leading to faster and more responsive deque performance.
2. **Lower Overhead than MCS Locks in Short-Duration Lock Scenarios**:
   - Deque operations are typically quick, involving brief locking periods. The MCS lock, while effective in fair queuing, adds overhead due to its queue management. The TTAS lock's simpler structure and lower overhead make it more suitable for deques, where the lock is held for a short duration, ensuring quicker processing of deque operations.
3. **Backoff Strategy Minimizes Contention Impact**:
   - In multi-threaded environments, deques often face simultaneous access at both ends. The backoff strategy in TTAS locks, where threads wait for a randomized period after a failed lock attempt, effectively reduces contention. This approach is particularly beneficial for deques, ensuring smoother and faster access even under frequent and concurrent operations, a common scenario in many applications utilizing deques.

4. **Scalability in High-Contention Environments**:
  - As the use of a deque scales, particularly in applications like task scheduling or buffering, it may experience high contention. The scalability of TTAS locks, due to their reduced bus traffic and simplicity, ensures that the performance of the deque remains robust under heavy loads. This scalability is essential for maintaining high throughput and responsiveness in systems where deques play a critical role in data processing.

In conclusion, the TTAS backoff lock, with its reduced bus traffic, lower overhead for short-duration operations, effective backoff strategy in contention management, and scalability in high-contention environments, offers a tailored and efficient locking mechanism for deques. These characteristics make it a particularly suitable choice for enhancing the performance and efficiency of lock-based deques in various demanding and concurrent operational scenarios.



Number of Elements in Fibonacci series: 496
Number of Elements in Qsort: 100

Lock-Free Deque:

Lock-Free using CAS:
Implementing a CAS (Compare-And-Swap)-based lock-free deque offers several advantages over other lock-free deques, particularly under conditions of high contention. The core strength of this approach, as detailed in Maged M. Michael's paper "CAS-Based Lock-Free Algorithm for Shared Deques," lies in its efficient handling of concurrent operations without the need for locks.

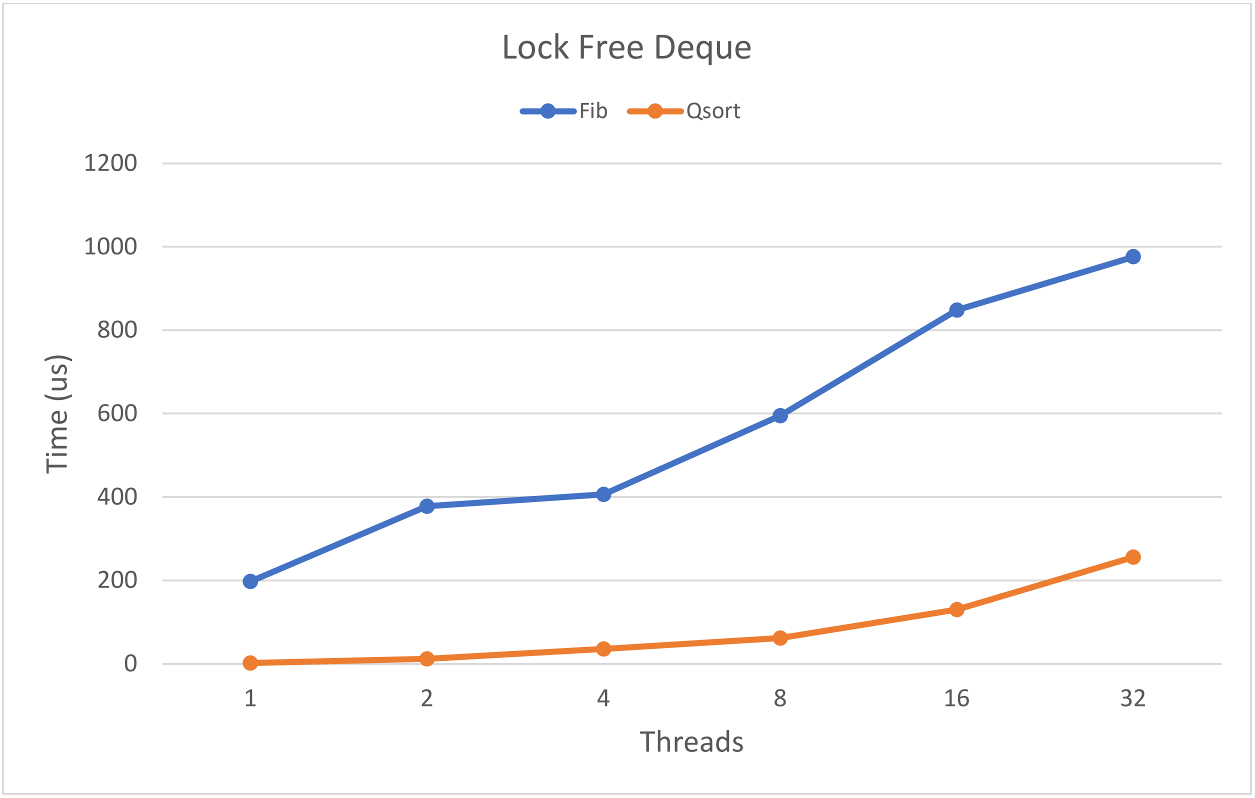
Firstly, the CAS-based approach significantly reduces the overhead associated with locking mechanisms. Locking can lead to performance bottlenecks, especially under heavy load, as threads wait for access to the deque. The CAS-based deque, in contrast, allows multiple threads to operate on different ends of the deque simultaneously without waiting, thereby increasing throughput.

Secondly, the CAS operation provides a simple yet powerful way to ensure the atomicity of updates. This is crucial for maintaining the consistency of the deque in a multi-threaded environment. The lock-free design ensures that even if a thread is suspended mid-operation, it doesn't block other threads, unlike lock-based approaches where a locked resource can lead to significant delays.

Moreover, the algorithm's lock-free nature also eliminates the possibilities of deadlocks and priority inversion problems, which are common in lock-based systems. This makes the system more robust and reliable, particularly in systems where high availability is critical.

Finally, the scalability of the CAS-based deque is superior. As the number of threads increases, the performance of lock-based deques often degrades due to increased contention for locks. In contrast, the CAS-based deque scales more gracefully with the number of threads, making it ideal for systems with high parallelism.

In summary, a CAS-based lock-free deque, as proposed by Maged M. Michael, offers enhanced performance, better scalability, and greater robustness under high contention scenarios compared to other lock-based deques.



Number of Elements in Fibonacci series: 496
Number of Elements in Qsort: 100