Name: Tanisha Rana
Section: CSE4
Roll No.: SE20UCSE202

README

1) TAS Lock

This Java program is an implementation of a Test-and-Set (TAS) lock, which is a type of spinlock. The program creates multiple threads to increment a shared counter variable. The TAS lock is used to ensure that only one thread can access and modify the counter variable at any given time. Here are some key points about the code:

- The program imports the AtomicBoolean class from the java.util.concurrent.atomic package. This class provides atomic operations on boolean values, which are used to implement the TAS lock.
- The program defines a TASLock class that has two methods: lock() and unlock(). The lock() method sets the state of the AtomicBoolean object to true using the getAndSet() method, which returns the previous value of the object before setting it to true. This creates a busy wait loop (spin) that waits until the previous value of the object is false (i.e., the lock is not currently held). The unlock() method simply sets the state of the AtomicBoolean object to false.
- The program defines a main() method that creates a specified number of threads and assigns them to increment the shared counter variable a specified number of times. The number of iterations per thread is calculated based on the total number of iterations (5,000,000) and the number of threads. The program uses the start() method to launch each thread and the join() method to wait for all threads to complete before exiting.
- The program calculates the total execution time and prints out the final value of the counter variable, the throughput (i.e., the number of operations per second), and the total number of operations. The throughput is calculated by dividing the total number of operations by the total execution time in seconds.

Overall, this program provides an example of how to implement a spinlock using the TAS algorithm and demonstrates how spinlocks can be used to synchronize access to shared resources in a multithreaded environment.

2) TTAS Lock

This Java program is an implementation of a Test-and-Test-and-Set (TTAS) lock, which is another type of spinlock. The program creates multiple threads to increment a shared counter variable. The TTAS lock is used to ensure that only one thread can access and modify the counter variable at any given time. Here are some key points about the code:

- The program imports the AtomicBoolean class from the java.util.concurrent.atomic package. This class provides atomic operations on boolean values, which are used to implement the TTAS lock.
- The program defines a TTASLock class that has two methods: lock() and unlock(). The lock() method first checks if the lock is already held by another thread. If it is, the method enters a busy wait loop (spin) that waits until the lock becomes available. If the lock is not currently held, the method attempts to acquire the lock by setting the state of the AtomicBoolean object to true using the getAndSet() method. If the previous value of the object was false (i.e., the lock was not already held), the method returns, indicating that the lock has been

acquired. If the previous value of the object was true, the method continues spinning until the lock becomes available.

- The program defines a main() method that creates a specified number of threads and assigns them to increment the shared counter variable a specified number of times. The number of iterations per thread is calculated based on the total number of iterations (5,000,000) and the number of threads. The program uses the start() method to launch each thread and the join() method to wait for all threads to complete before exiting.
- The program calculates the total execution time and prints out the final value of the counter variable, the throughput (i.e., the number of operations per second), and the total number of operations. The throughput is calculated by dividing the total number of operations by the total execution time in seconds.

Overall, this program provides an example of how to implement a spinlock using the TTAS algorithm and demonstrates how spinlocks can be used to synchronize access to shared resources in a multithreaded environment. The TTAS algorithm is similar to the TAS algorithm, but it includes an additional optimization that reduces the number of atomic memory accesses required to acquire the lock.

3) TTAS with Backoff

This code implements a multi-threaded program that increments a shared counter variable using the test-and-test-and-set (TTAS) algorithm with backoff. The program creates a specified number of threads and each thread increments the counter a specified number of times. The program uses a TTASLock class, which provides methods for acquiring and releasing the lock. The Backoff class is used to implement an exponential backoff algorithm, which is used when the lock is not available.

The main method initializes the shared counter variable and creates the specified number of threads. The incrementCounter method contains the logic for incrementing the shared counter variable using the TTAS algorithm with backoff. The TTASLock class provides methods for acquiring and releasing the lock. The Backoff class is used to implement an exponential backoff algorithm, which is used when the lock is not available.

At the end of the program, the main method prints the final value of the counter variable and the program's throughput in operations per second.