**Programming Assignment 2**

**Implementation of Filter Lock Algorithm**

**and**

**Peterson Tree Algorithm**

Name : Rohit Kapoor

Roll No: cs21mtech12011

**Program Design**

In the given program std::thread() function has been used for multithreading therefore it requires the programmer to explicitly link the lpthread file for the program to work correctly.

In the “output-log.txt” file the timestamps of the entry and exit calls are entered for each such call performed.

For the Lock class the lock and unlock methods have been declared as virtual so that they can be overridden in the corresponding Filter and peterson classes.

**Peterson Tree Lock Algorithm**

Peterson tree locking is a mechanism for achieving mutual exclusion, it is applicable only in the cases where the number of threads is a power of 2. This approach requires the creation of a binary tree of peterson locks, now the peterson lock is able to achieve mutual exclusion for 2 threads per lock. For this implementation since the number of threads is more than 2, the peterson lock algorithm has been modified to contain to contain an array of n threads and also the flag variable has been modified for n threads.

By creating a binary tree of these locks, certain levels are created and only a limited number of threads can enter each level of the tree i.e. each level reduces the number of threads that can enter it by half. Since it is implemented for a power of 2, therefore it can be assumed that a full binary tree is achieved.

Critical section

n/2 threads

n threads

At each level of the tree the number of threads that can enter is reduced by half and only one thread can enter in to the critical section.

When the threads obtain the locks, the locks are acquired starting from the leaves all the way to the root node. And when the locks have to be released then the unlock operations happen in the opposite order i.e. starting from the root node till the leaf possitions.

All the threads might get mapped to a single leaf node, in order to solve this issue, the thread id is used to calculate to possition i.e. n/2 + id/2.

**Filter Lock Algorithm**

The filter lock algorithm implementation is same as that discussed in the class. N levels are maintained in the bucket at each level one of the threads is choosen as a victim and it cannot enter into the subsequent levels unless there are other threads in the next level.

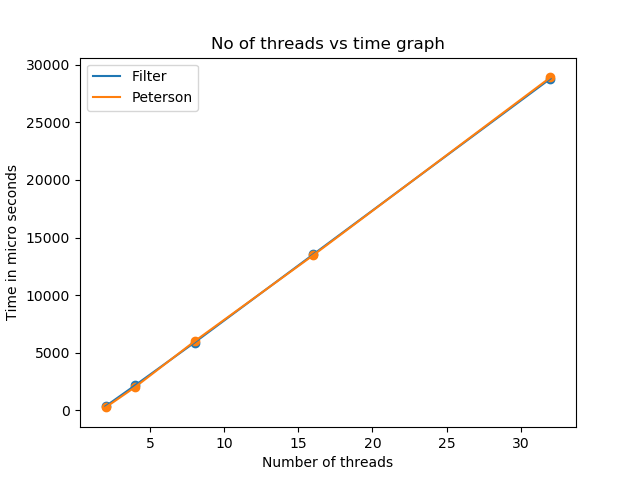
-- The thread needs to wait if there is anyother thread in the subsequent levels.

-- Only one thread is able to enter the cs at any point of time other threads can enter the cs section once the first thread has completed its operations.

**Plots**

**Average Time required to enter the Critical section vs the number of threads.**

The plots have been constructed by varying the number of threads from 2 to 64 in the powers of 2 and the parameter k (i.e. the no of times a thread enters the cs) has been fixed to 10.

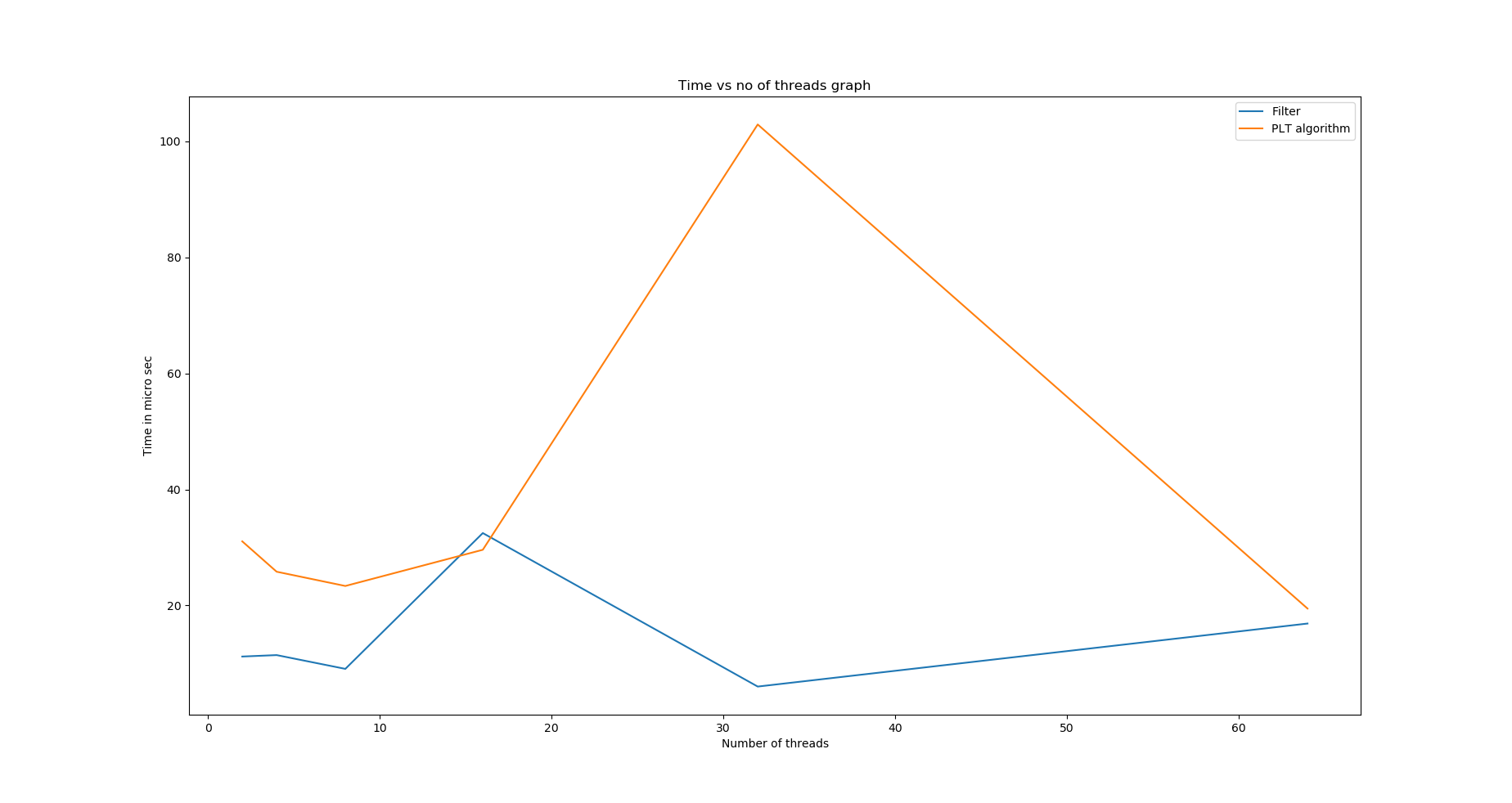


- The average time taken to enter the critical section is almost the same for both the algorithms with slight differences.

- At certain values peterson tree algorithm is performing better while at other values Filter algorithm is performing better but the differences are very subtle.

**Average Time taken to Exit the critical section vs the no of threads**

The no of threads vary from 2 to 64 in the powers of 2 and the value of k remains fixed at 10.



- Since the unlock() function consists of just one operation in case of filter algorithm, hence it has lower values while the unlock function consits of multiple operations in peterson tree algorithm hence it is taking more time.