Operating Systems II : CS3523
Spring 2019

Lab Exam : Implementing Class Mutual Exclusion

Instructor : Dr. Sathya Peri
Author : Puneet Mangla (CS17BTECH11029)

Design of Program :

I implemented the class mutual exclusion problem using conditional variables in c++.

Condition variables

Condition variables are synchronization primitives that enable threads to wait until a particular condition occurs. Condition variables enable threads to atomically release a lock and enter the sleeping state.

When you want to sleep a thread, condition variable can be used. In C under Linux, there is a function `pthread_cond_wait()` to wait or sleep.
On the other hand, there is a function `pthread_cond_signal()` to wake up sleeping or waiting thread.

Function definitions:

```
int pthread_cond_wait(pthread_cond_t *restrict cond,
                      pthread_mutex_t *restrict mutex);

int pthread_cond_signal(pthread_cond_t *cond);
```

Where ,

`cond` : condition variable

`mutex` : is mutex lock

pthread_cond_t condition

condition variables to keep the processes waiting which dont belong to current session.

pthread_mutex_t mutex

mutex variable to be passed in condition variable.

int curr_class

current session of program.

void entry_sec (int s)

```
pthread_mutex_lock(&mutex); // aqquire lock first
if (entry_count>0 && s!=curr_class) {
    pthread_cond_wait(&condition, &mutex);
}
entry_count++;
if(entry_count==1)
    curr_class = s;
pthread_mutex_unlock(&mutex); // release mutex .
```

void exit_sec ( )

```
pthread_mutex_lock(&mutex); // aqquire lock
entry_count--;
if(entry_count==0)
    pthread_cond_signal(&condition);
pthread_mutex_unlock(&mutex); // release lock
```

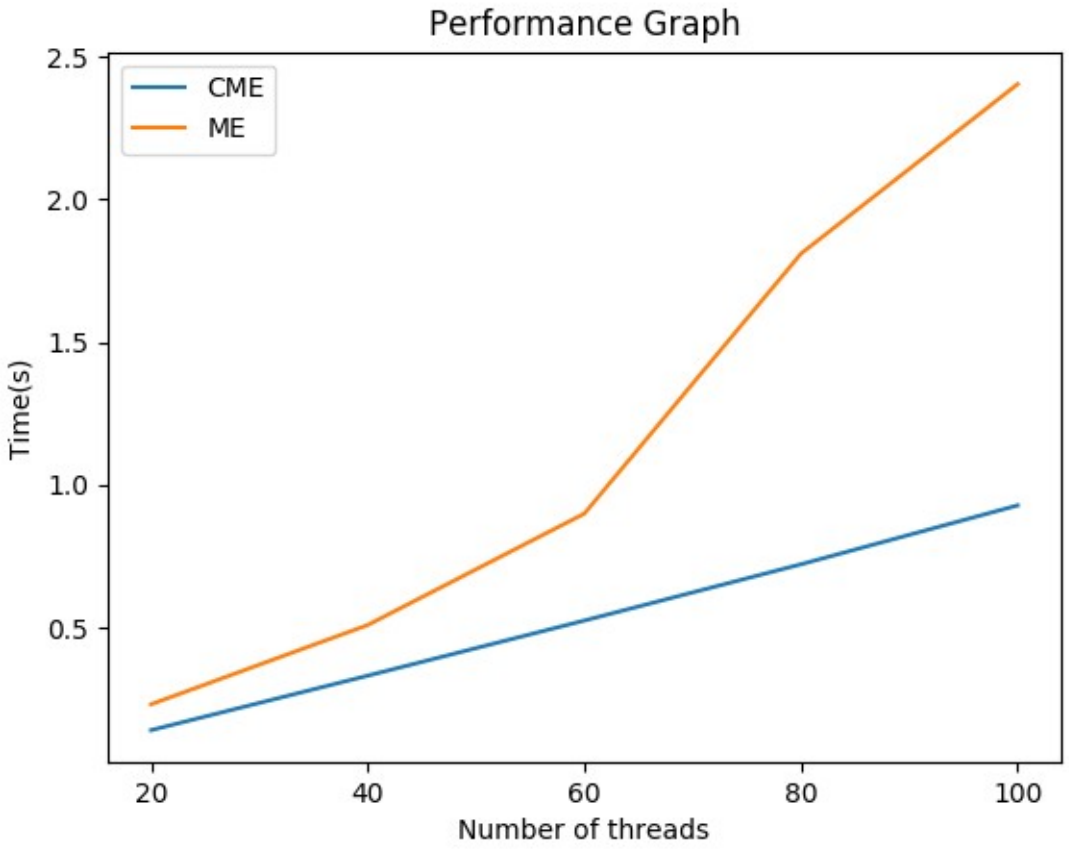
Comparison metrics :

1. **Average request waiting time :** the average time taken by sheduler to process a request method made by a thread.

Graphs :

For comparison we drew a graph with the average request times taken by each thread to enter the CS in the y- axis frequency in x-axis. Let the number of threads vary from 20 to 100 in the increments of 20 on the x-axis. We compared it against ME and CME algorithm. Fix all the other parameters as:

$$n = 20,\ S = 100,\ \mu_{CS} = 0.01s,\ \mu_{rem} = 0.05s$$



From above graph we can see that SME algorithm always takes larger time than CME algorithm. This is because in CME many threads can enter CS because they can have same session whereas in SME two threads can’ t enter at same time.