Operating Systems II : CS3523
Spring 2019

Programming Assignment 5 : Implement Dining Philosopher's using Conditional Variables

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

Design of Program :

We implemented the dining philosophers 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

enum state

```
enum state {THINKING, HUNGRY, EATING};
```

represents the possible states of a philosopher.

pthread_cond_t *condition

array of n condition variables for each philosopher.

pthread_mutex_t mutex

mutex variable to be passed in condition variable.

enum state *status

status of all philosopher at a given time.

void pickup (int i)

```
pthread_mutex_lock(&mutex);
status[i] = HUNGRY;
test(i);
if (status[i] != EATING) {
    pthread_cond_wait(&condition[i], &mutex);
}
pthread_mutex_unlock(&mutex);
```

void putdown (int i)

```
pthread_mutex_lock(&mutex);
status[i] = THINKING;
test((i+1)%n);
test((i+n-1)%n);
pthread_mutex_unlock(&mutex);
```

Comparison metrics :

1. **Average waiting time :** the average time taken by a thread to enter the CS.
2. **Worst Case waiting time** : the maximum time taken by a thread to enter the CS.

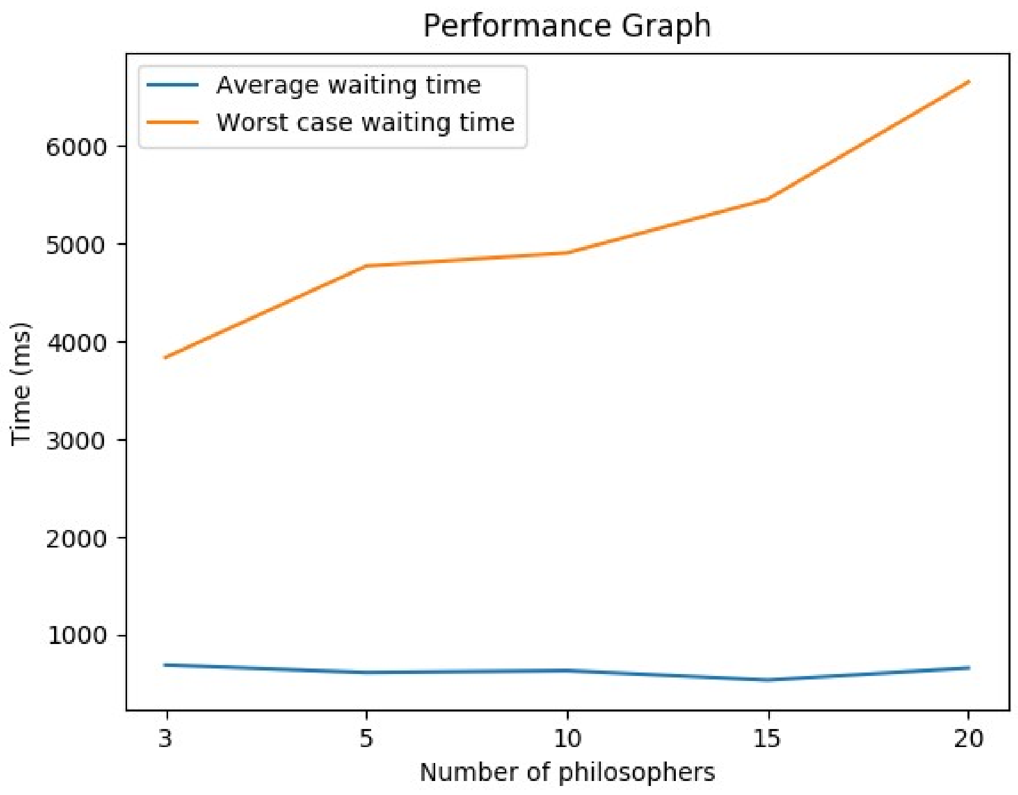
Algorithm :

1. The main thread reads the "inp-params.txt" and stores all the parameters as global for all theads to be accessible.
2. n philosopher threads are created and their id are stored in array pthread_t tid[n_w] .
3. default_random_engine is used to generate random numbers from exponential_distribution to stimulate critical and remainder section . Parameters to the distribution were were $1/\mu_{cs}$ and $1/\mu_{rem}$ respectively.
4. Initial status of all philosophers is set to THINKING.
5. void* philosopher function stimulate the working of philosopher.
6. void pickup function uses conditional variables to check if the philosopher can eat or not. If condition is failed than corresponding lock is released and the thread is blocked by the corresponding condition variable. If condition is satisfied lock is aqquired , thread is allowed to eat and lock is released afterwards.
7. void putdown function notifies the adjacent philosophers that chopstick is free now. Status of calling philosopher is set to THINKING.
8. For printing logs fprintf/printf are used instead of cout as cout streams causing mixing of logs.

Graphs :

For comparison we drew a graph with the average and worst times taken by each thread to enter the CS in the y-axis and the number of philosopher threads n varying in x-axis. Let the number of the philosopher threads n vary from 1 to 20 in the increments of 5 on the x-axis. Fix all the other parameters as:

$$h = 10,\ \mu_{CS} = 1,\ \mu_{rem} = 2$$



From above graph we can see that the worst case waiting time increases monotonically with number of philosophers because as the number of philosophers increases the less threads wil be waiting in respective condition queue. Now since there are lot of threads and only one mutex lock they suffers from starvation.

The average waiting time remains almost constant . This is because average waiting time for every thread is dependent only on its two neighbours which remains almost same when n is sufficiently large.

Conclusion :

We can use conditional variables to implement problems that can be solved using monitors.

In the problem of dining philosophers problem the worst case time increases monotonically with number of philosophers but the average time almost remains constant.