Mohan Konyala

UFID: 5005-3565

Lab 01 Report

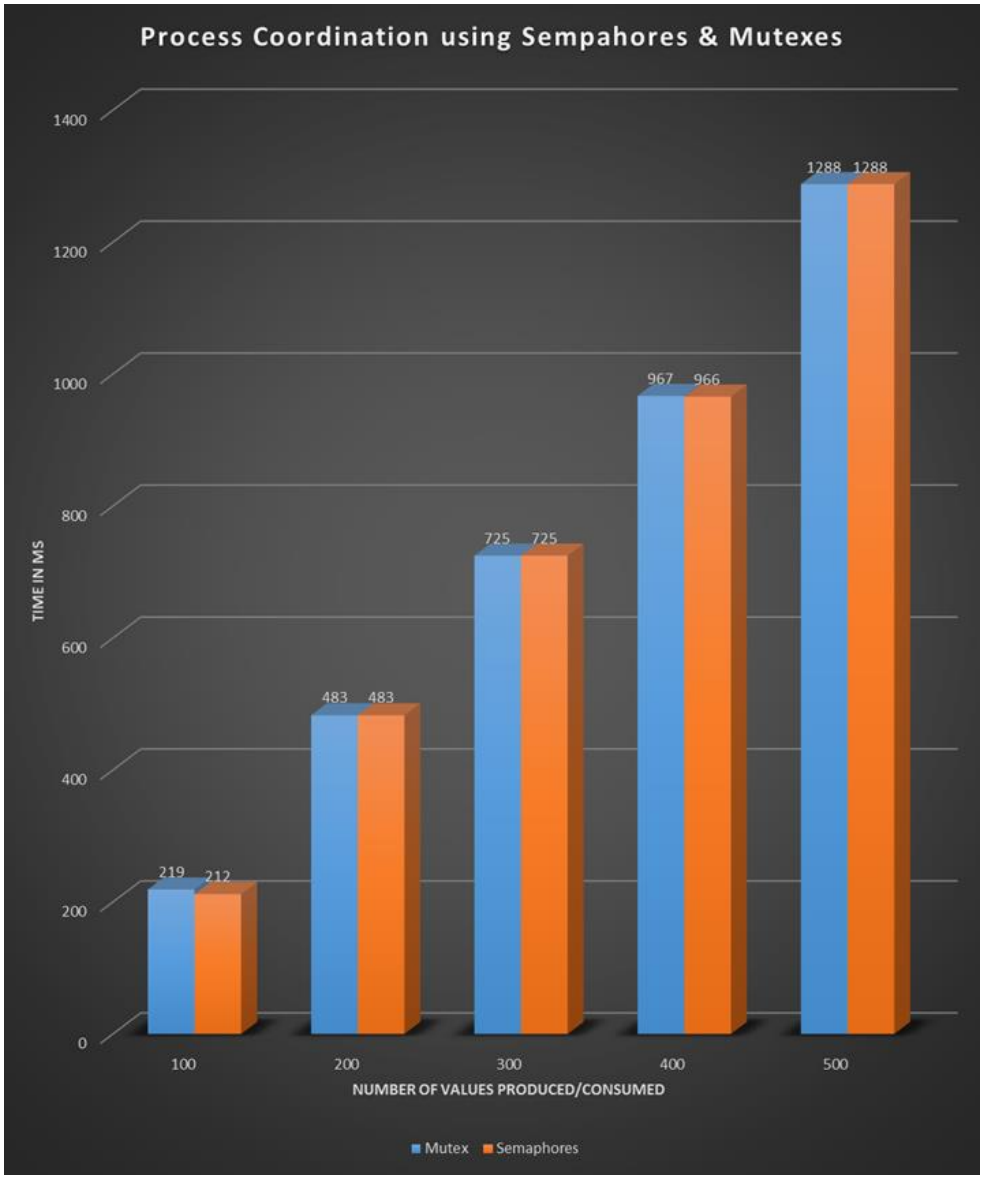
A) When the two processes share access to the same buffer and run at different speeds the main problem would be the producer producing at higher rate than the rate at which consumer can consume and vice versa.

If the producer is inserting elements into the buffer at higher speed than the consumer can consume the buffer eventually gets full and some data would be overridden on the buffer before the consumer consumes it. Similarly, if the consumer consumes at higher rate than the producer can produce, at a particular time the buffer would be empty and the consumer will be consuming empty values.

A synchronization procedure between the producer and consumer is better way to solve the above mentioned issues. The producer should wait for the consumer to consume value and indicate the producer that it has consumed so that producer continues with producing more values. The same way consumer should wait for producer to produce the value and indicate it that the value has been produced and it can carry it consume action. If the buffer is full the producer should be kept in wait state until consumer consumes a value and if the buffer is empty consumer should be wait until producer produces a value

The producer should wake up the consumer process after producing value else the buffer eventually gets full and producer will be waiting infinitely for the consumer to consume the values. The situations like Dead lock might also occur where both the producer and consumer will acquire half the resources and each keep waiting for the other half which are acquired by other process and never going to be released until both of them complete execution which will never happen.

Timing Graph for (C) and (D)



E) The implementation with mutex makes the critical section code available to either producer or consumer at a given time. The producer produces a value and signals consumer to consume it. If the consumer tries to consume while the producer is producing it has to wait. If producer tries to produce while consumer is consuming it has to wait. The production and consumption actions are alternate here producing/consuming a single value before going to wait.

Coming to the implementation using mutex and semaphores, depending on the speed at which the producer can produce it will fill the buffer with multiple values before made to wait. Similarly, the consumer can consume multiple values from the buffer before the processor switch to producer process.

And also, if the producer works faster and fill the buffer completely, then it is made wait so that the consumer can consume and make empty slots to fill in values again. If consumer consumes faster and the buffer is empty, then consumer is made to wait so that it won't read empty values and the producer can fill in the values.

F) Conclusion

From the above graph, we can conclude that the producer-consumer problem implemented with semaphores and mutex (problem d) has faster running time than implementation only using mutex (problem c). This is because in (c) the value is produced then the value is consumed i.e., producer and consumer must wait alternatively to perform their action. To produce 'n' number both producer and consumer will be made wait n times. This is inefficient as the other process has to definitely wait to perform further action.

In (d), the producer can produce multiple values before it is made to wait, similarly, the consumer can consume multiple values which results in reduced overall waiting time and faster running time. Though, the time difference is few milliseconds which is trivial, for large enough production/consumption the time difference would be huge.