BILKENT UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CS342
Operating Systems

Project 3
Synchronization and Deadlocks

Ahmet Ayrancıoğlu
21601206

Results of the Experiments

Handling Method	Deadlock Nothing	Deadlock Avoidance	
Process Count	**Exec time**	**Exec time**	**Difference**
5	0.058	0.061	5%
10	0.123	0.141	14%
15	0.182	0.218	19%
20	0.253	0.337	33%

Table 1

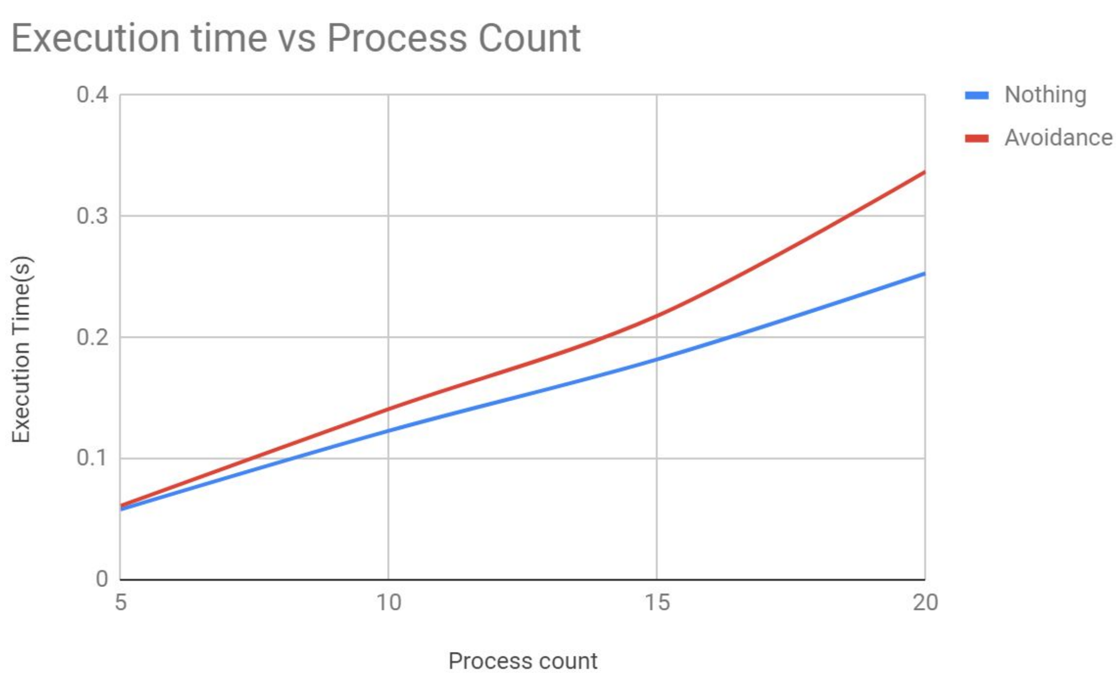

Figure 1 - Execution time vs Process Count

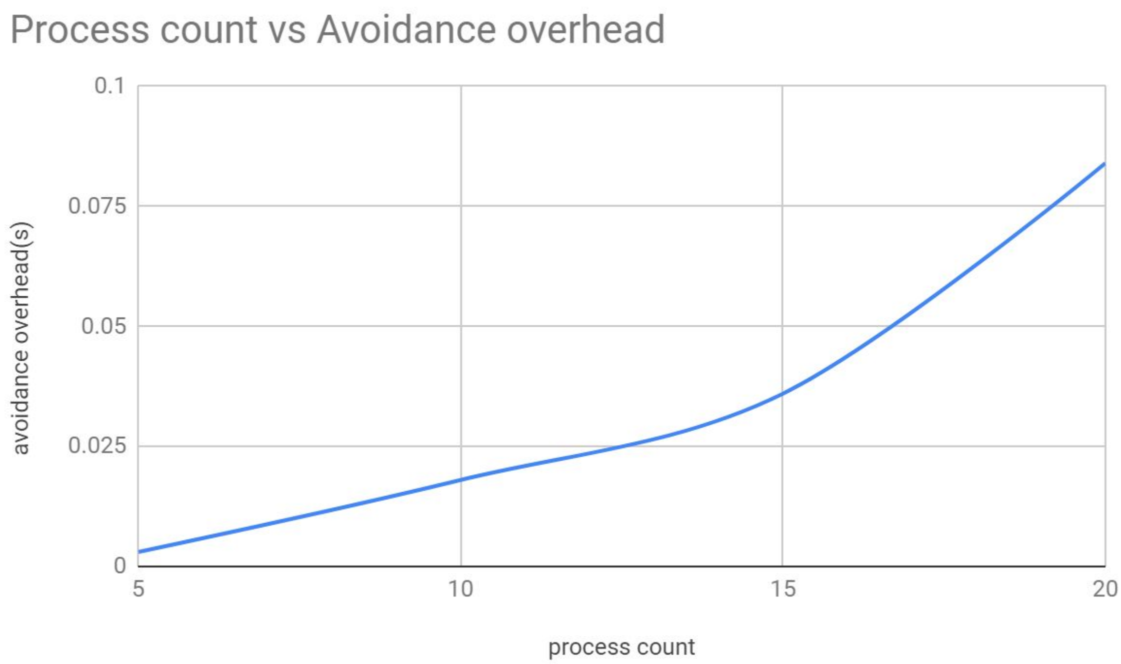

Figure 2 - Process count vs Avoidance overhead

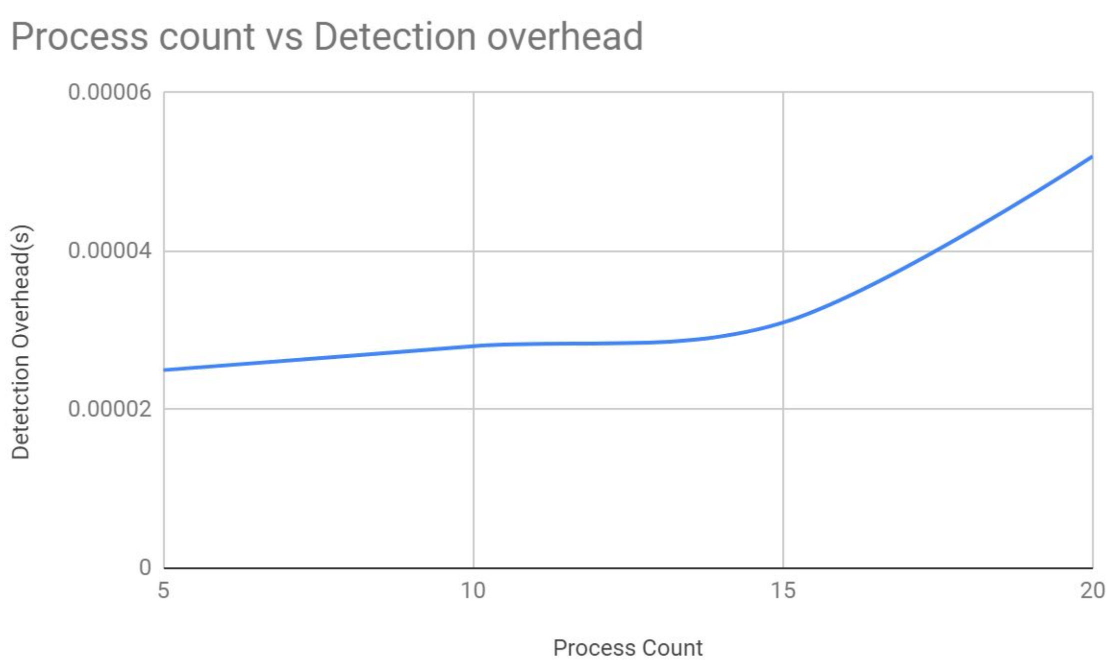

Figure 3 - Process count vs Detection overhead

Interpretation and Conclusion

Cost of Deadlock Avoidance

These results shown in Table 1, Figure 1 and Figure 2 were collected using a modified version of the example app.c. In the experiment, each thread requests 2 of each resource twice back to back and then releases all of its resources and does this inside a for loop for 100 times. This experiment was done 4 times with different thread counts and existing resource count was adjusted so that the system did not go into deadlock when running deadlock nothing.

To avoid deadlocks, each request is tested against the banker's algorithm to check if the request can cause a deadlock or not. This process makes sure that a thread can not go into deadlock but the algorithm is an expensive one to run so it causes a performance overhead. As shown in Figure 1, as the number of processes increases, the difference in execution times between deadlock avoidance method and the deadlock nothing method also increases. This is expected as the banker's algorithm tries to find a safe sequence and as the number of processes increases the time it takes the algorithm to find a safe state also increases. This increase in the performance overhead is clearly shown in Figure 2.

Cost of Deadlock Detection

The cost of deadlock detection is very similar to the cost of deadlock avoidance as both algorithms are quite similar. So, interpretations from the first part of the report can be applied to deadlock detection as well. This is also supported in Figure 3, as it can be seen that the performance overhead is again increasing as the number of processes increase.

The results shown in figure 3 was collected by measuring the time of single deadlock detection and was repeated for different process counts.