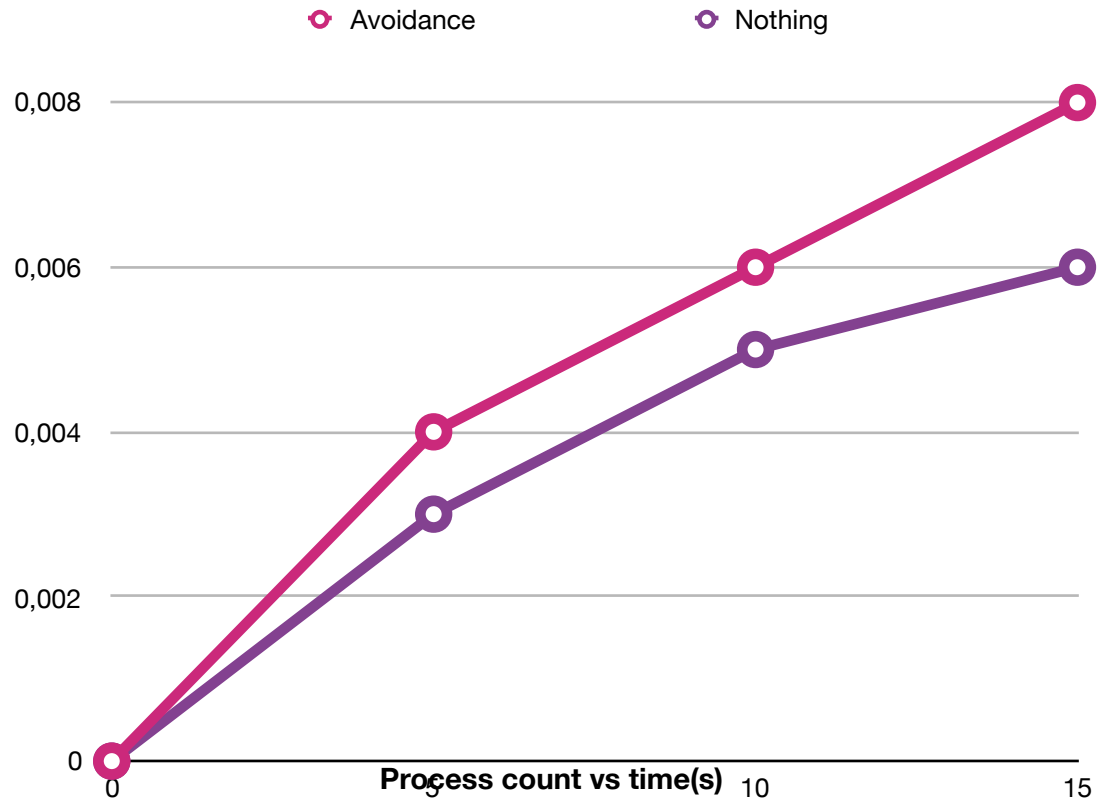
Berfin KÜÇÜK / 21502396
Ömer Faruk BABADEMEZ / 21601216

In this project, we have implemented resource allocation procedure between threads. To do that we have analyzed deadlock concept as well. In this regard, deadlock detection and avoidance mechanisms have been practiced along with synchronization in Phtreads API, semaphore method is used.

In deadlock avoidance method, while a process requesting resource(s), it is granted only when situation is safe after allocation and not granted if it is unsafe. In this regard, processes waiting for request if it is unsafe.

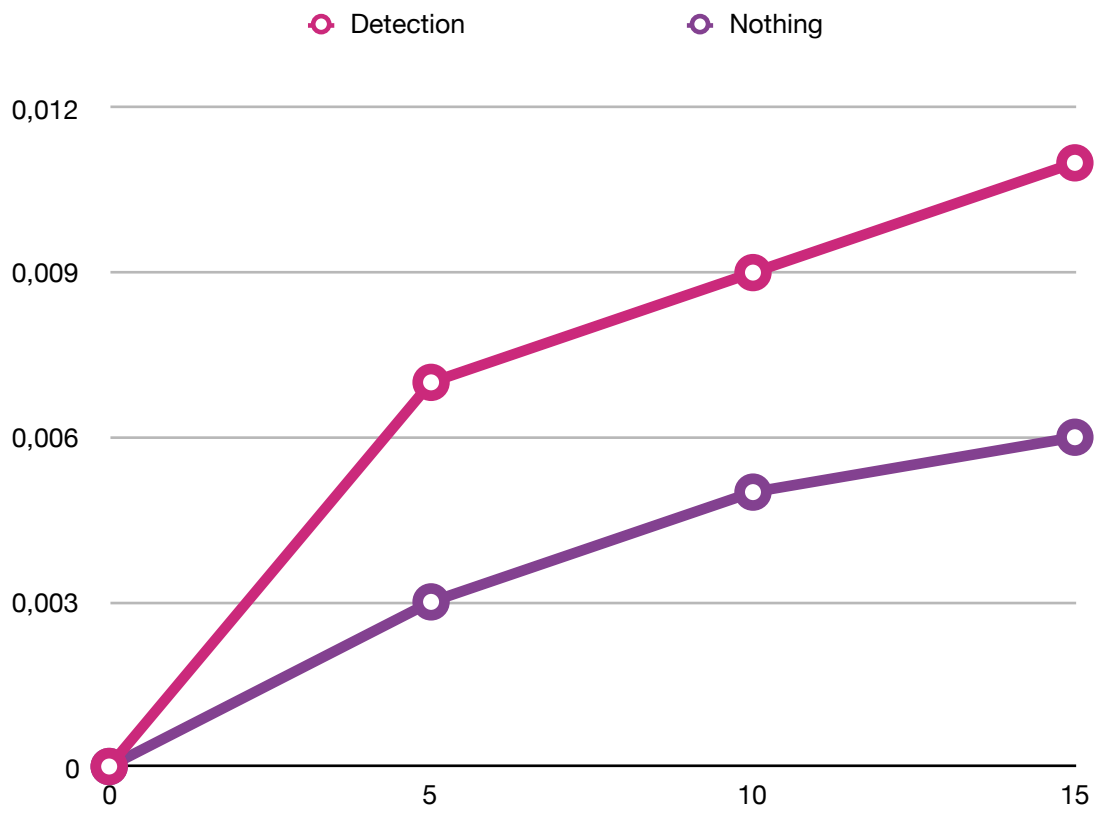
In order to conduct an experiment to test overhead of deadlock avoidance, app.c given by instructor is modified. First of all, unmodified code given cause only thread with pid 4 to enter *aprocess* function. Thus, sleep(1) is added to fix overhead of incrementing i, as shown below. In that regard, the file is executed with "time ./app " command to see time overhead and also since sleeping cause extra time overhead, the result of time will be decremented by thread number in analysis part.

```
for (i = 0; i < N; ++i) {
 pids[i] = i;
 pthread_create (&(tids[i]), NULL, (void *) &aprocess, (void *)&(pids[i]));
sleep(1); //modified part
}
```



The graph shown above is results of always requesting safe resources, hence it is never deadlocked when DEADLOCK_NOTHING is used. Thus, extra cost of evaluating if state is safe result in time overhead as seen above. However, if there would be a unsafe state allocation along with a deadlocked state then time of deadlock_nothing would be infinite which is dramatically worse than deadlock avoidance method. Hence, considering overall time efficiency of using deadlock avoidance is better than deadlock nothing method.

In deadlock detection method that we have implemented in this project does not consist any recovery method. It just detect whether there is deadlock or not. Moreover, in this method, requested resources are granted if there is available resource without checking whether it will be safe or unsafe. The experiment conducted for using or not using that method is similar to the experiment mentioned above.



Process count vs time(s)

Again, the experiment above is tested with always safe resource allocation because otherwise it might have infinite deadlocked state. As seen above, time passed in deadlock

detection is more than deadlock avoidance method because of extra deadlock checking in main method. If we wouldn't check it, it would be almost same time cost.

In conclusion, deadlock avoidance method is the best among all deadlock mechanisms because there is no possibility of deadlock which would result in infinite time cost. If states are always safe, time cost of deadlock_nothing method is the best among all mechanism. However, there is no such guarantee.