Emre Tarakcı – 21902607

Doruk Onur Çalışkan - 21902672

Operating Systems – CFS Scheduling Project Report

In this project, we have programmed a CFS scheduler in C. CFS stands for Completely Fair Scheduling and it is the default process scheduler that the Linux kernel uses. It has the concepts of virtual runtime and timeslices, which ensure that processes can get their share of the CPU based on their priorities. In our case, we simulated processes and the scheduler using threads.

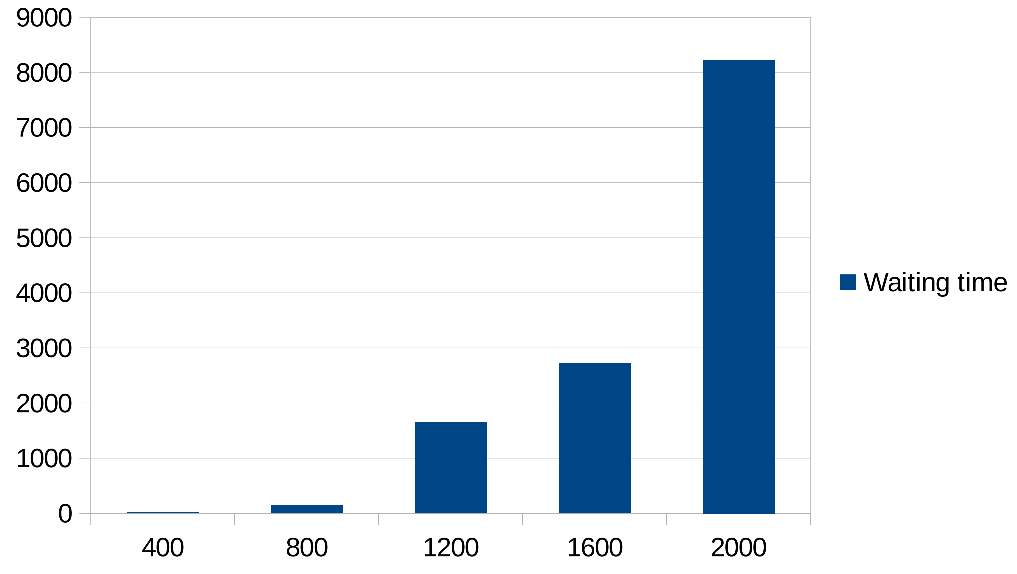
Our program has a generator thread that generates threads according to inter arrival time, a scheduler thread that signals process simulations to wake them up, the main thread from where the generator and scheduler threads are created and initializations are made and the process threads which simulate a running behavior using usleep command. To prevent race conditions, we have used mutex locks and to signal processes and the scheduler, we have used condition variables. Each process has their own condition variables, and the process threads also have their condition variable to sleep on, pointed at in their PCBs.

After we implemented the program, we made some experiments with our program to see the relation between the avgIAT, avgPL and the average wait time. The results of our experiment will be interpreted in the remainder of this report.

1) Average Program Length Versus Average Waiting Time

The average program length determines the length distribution of the generated processes along with the max program length and min program length. For our purposes, we fixed min program length as 100 and max program length as 2400. The results of our experiments, when the other parameters were fixed, resulted in an exponentially increasing graph.

minPrio	maxPrio	avgPL	minPL	maxPL	avgIAT	minIAT	maxIAT	rqLen	ALLP	Waiting time
-20	19	**400**	100	2400	1200	100	2400	30	30	**25**
-20	19	**800**	100	2400	1200	100	2400	30	30	**141**
-20	19	**1200**	100	2400	1200	100	2400	30	30	**1658**
-20	19	**1600**	100	2400	1200	100	2400	30	30	**2727**
-20	19	**2000**	100	2400	1200	100	2400	30	30	**8228**

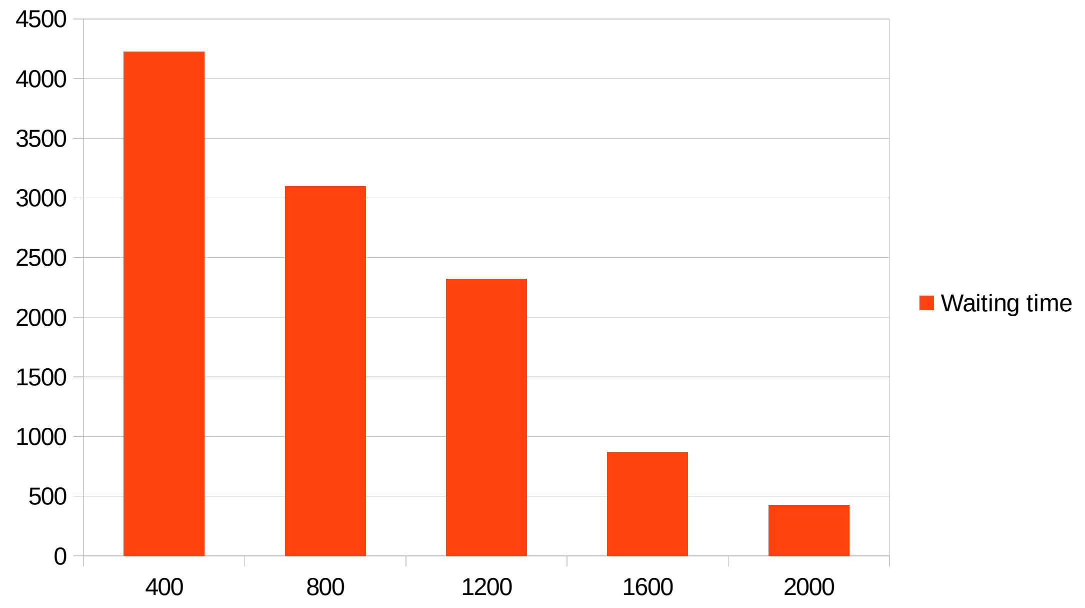


This happens because when the lengths of the processes increases, there are more processes that simultaneously exist in the run queue. This is also related to the inter arrival time, and in this case since it is fixed, the competition for the CPU increases as more processes simultaneously want to be executed due to the increased process lengths. In that case, if lower priority processes are in the queue along with the higher priority ones, the higher priority processes will have to wait for a much longer time, staying in the CPU for longer. This causes an increase in the waiting time.

2) Average Inter-Arrival Time Versus Average Wait Time

The average inter-arrival time determines how often new processes are created and added to the running queue. This value affects competition for the CPU, because if processes are being created very often, the queue gets more full, therefore more competition for the CPU. This leads to an increase in the average wait time. If the inter-arrival times increase while other parameters are fixed, that means the CPU has more time to spend on each one of the processes and the runqueue does not get as congested.

minPrio	maxPrio	avgPL	minPL	maxPL	avgIAT	minIAT	maxIAT	rqLen	ALLP	Waiting time
-20	19	1200	100	2400	**400**	100	2400	30	30	**4225**
-20	19	1200	100	2400	**800**	100	2400	30	30	**3097**
-20	19	1200	100	2400	**1200**	100	2400	30	30	**2322**
-20	19	1200	100	2400	**1600**	100	2400	30	30	**869**
-20	19	1200	100	2400	**2000**	100	2400	30	30	**427**

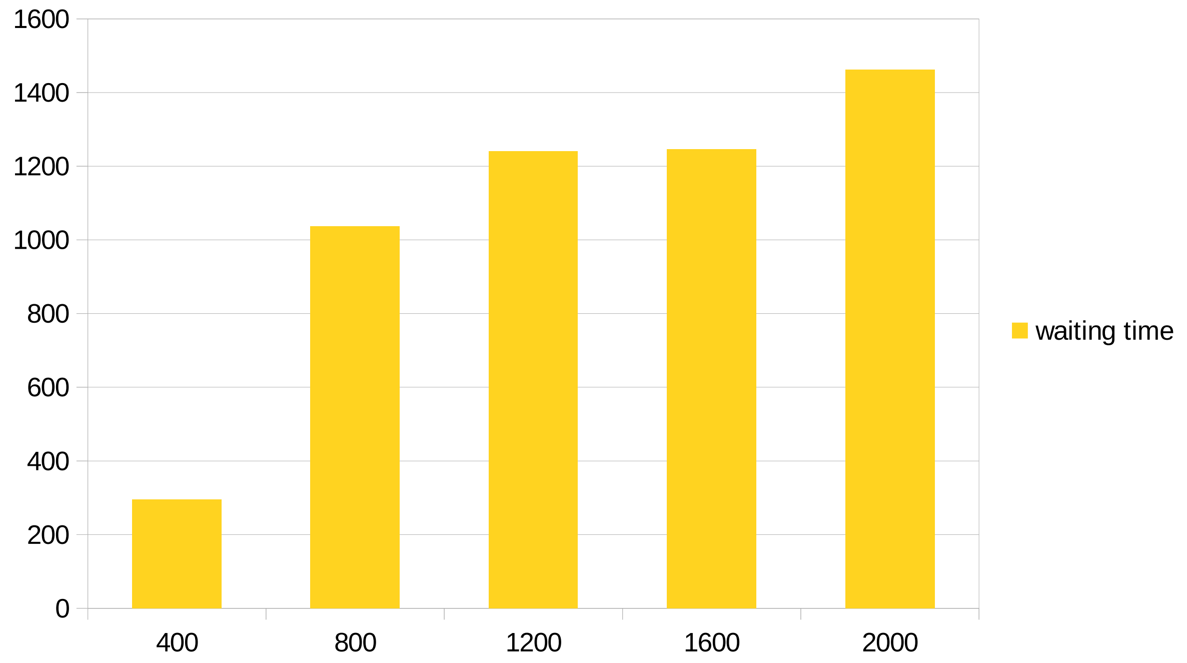


In the values, we can see that the waiting time values decrease as the inter-arrival time increases. This decrease seems to be more linear compared to the results of the previous graph. In fact, if the inter-arrival times are more than the length of the processes, we would expect there to be no waiting time at all in the scheduler, because the CPU would be able to complete the processes before the arrival of the next process, keeping the queue empty at all times.

3) Average Inter-Arrival Time and Average Process Length Increased Simultaneously

As we saw that average inter-arrival time decreases the waiting time and average process length increases it in exponential distribution, we wanted to test the results of increasing them both by the same amount. Our guess was that the waiting time would be close to zero, since we are increasing process lengths but giving the CPU more time by increasing the inter-arrival times as well.

minPrio	maxPrio	avgPL	minPL	maxPL	avgIAT	minIAT	maxIAT	rqLen	ALLP	Waiting time
-20	19	**400**	100	2400	**400**	100	2400	30	30	**296**
-20	19	**800**	100	2400	**800**	100	2400	30	30	**1037**
-20	19	**1200**	100	2400	**1200**	100	2400	30	30	**1240**
-20	19	**1600**	100	2400	**1600**	100	2400	30	30	**1246**
-20	19	**2000**	100	2400	**2000**	100	2400	30	30	**1461**



The results of this experiment did not yield results that we previously expected. Since the values are not fixed and an exponential distribution is used, increasing them both at the same amount would not make the waiting times zero. This is because, if we used fixed values, that could be the case, excluding the overhead times. The reason why there is still an increase in the wait time is because the inter-arrival times' lengths and process lengths are not synchronized. Meaning, the CPU can stay idle for a while due to a high inter-arrival time, then there may be two processes created with a low inter-arrival time. That would result in a waiting time in the scheduler, that is why when the inter-arrival times and the process lengths were increased simultaneously, the result did not yield waiting time close to zero.

Conclusion

To sum up, we have implemented a CFS scheduler in C using threads for simulation of processes, the scheduler and the generator. With our scheduler, we have run some experiments with changing average inter-arrival times, average process lengths and both of them simultaneously. We saw that increasing inter-arrival times decrease the wait time, increasing average process lengths increases average wait time, and increasing both of them causes a slow increase in the waiting times.