Caelan Mayberry & Justin Sybrandt

What does it mean?

We used the original test input using some additional processes for each scheduler. These inputs covered having two processes starting at the same time as well as one process arriving in the middle of another process. This gave us a few insights into the functionality of each scheduler.

Something we noticed was that the per core schedulers tended to be inconsistent in their output. For instance, FCFS – Single is consistent, rerunning the same process multiple times produced the same result, with zero wait times for every process. However, in FCFS – Per Core scheduler the processes sometimes had zero for wait times across the entire board of processes, and at other times a process would have a two second wait time due to the random assignment, which was both slower than the aforementioned time and FCFS – Single Core. RR – Per Core also suffered from a similar fate. It would have inconsistent results, while the RR – Load remained consistent.

What caused this was the scheduler would randomly schedule two processes to the same core. We found this happened most often because two of the processes arrived at the same time. This often increased wait times and would produce less efficient results.

We tested input where multiple processes of the same length came in at the same time at the end of the process. In the case of the two consistent schedulers, FCFS – Single and RR – Load, we found that the FCFS – Single was more efficient. However in the case where the same processes came in immediately, RR – Load’s efficiency decreased, but FCFS – Single’s efficiency got slightly better. In this case while the turnaround time increased by one second, but the wait time decreased by 3 seconds. These results can be seen by comparing the output of input1.txt and input6.txt. In these cases FCFS – Single proved to be more efficient overall.