|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CPUS | RUN TIME | WAIT TIME | SLEEP TIME |
| PRIORITY | 1 | 15 | 118 | 40 |
| 2 | 15 | 47 | 77 |
| 3 | 18 | 26 | 88 |
| RR | 1 | 14 | 112 | 43 |
| 2 | 16 | 52 | 74 |
| 3 | 21 | 34 | 83 |
| FCFS | 1 | 14 | 110 | 44 |
| 2 | 17 | 53 | 73 |
| 3 | 21 | 26 | 89 |

a.

Yes, the results are what I would expect. As we can see, the number of CPUs increasees, the average wait time decreases across all schedulers. Moreover, the PRIORITY scheduler consistently provided lower wait times than RR and FCFS as CPUs increase, which allign with its goals of favoring higher-priority processes.

b.

As we add more CPUs, the average wait time drops because more processes can run at the same time. This means they don’t have to sit around in the RUNNABLE state waiting for their turn. It can seen clearly in the numbers: wait times go from 118 → 47 → 26 for PRIORITY, 112 → 52 → 34 for RR, and 110 → 53 → 26 for FCFS. The PRIORITY scheduler does the best here since it gives preference to higher-priority processes, letting them run sooner. RR is fair and gives everyone a turn, but that also means it doesn’t prioritize urgent tasks. FCFS can get stuck when a long process is first in line, making others wait — though this problem gets better with more CPUs. Also, sleep time increases with more CPUs, because I/O-bound processes get picked faster and actually spend more time sleeping, just like they’re supposed to.