Process Scheduling

Andrew Harter
harteraj@uwec.edu

Abstract

With technological advances occurring faster than users have ever seen before, many of the background implementations that make these technological machines run are overlooked by the average users. One of the main oversights is within the operating system that is vital for these machines to run. One underlying question with regards to operating systems is what processes should be scheduled to execute and which ones should wait. There are many different ways to answerer these questions: "Should the scheduler be fair?"; "Should the scheduler sacrifice fairness for a more efficiency?"; "Will system applications run with a greater priority than user applications?" Unfortunately there is no correct or simple answer to this reoccurring question. Every scenario has its own benefits and drawbacks for scheduling different applications and in different environments. Due to these ever changing characteristics, one can only compare the difference in scheduling algorithms to one another with identical applications waiting to execute.

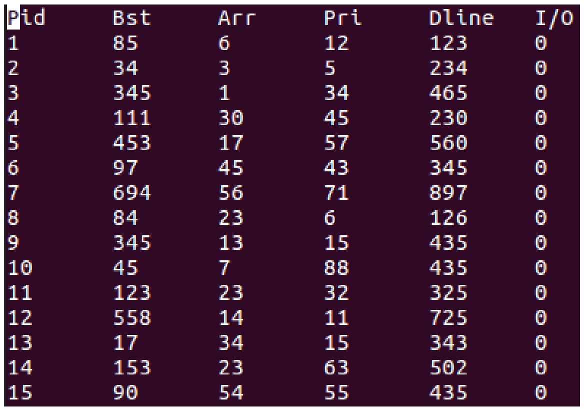
1. Introduction

The theories and techniques behind process scheduling are by no means new concepts. Examples of schedulers include: first come first serve, shortest job first, real time (process with the closest deadline executes first), round robin (every process executes for the same amount of time), feedback queues and hybrid schedulers. As stated previously, each scheduler has its own benefits and drawbacks. Processes will need to wait for others to have time to complete. One way to judge if one scheduler is more efficient than another in a given circumstance is by comparing the amount of time a process is waiting to execute.

To compare the algorithms, there are some scheduling statistics that are necessary to compare:

- total number of processes that are scheduled;
- clock tick (unit to measure process run-time);
- burst (amount of time for a process to execute);
- quantum (time allotted for process to run);
- context switching (any time a new process enters execution stage);
- average wait time (Average time spent by all processes not executing);
- turn-around time (lifespan of the process including time executing and waiting);

For this simulation of process scheduling, algorithms will be compared using multi-level feedback queue and real time scheduling. For simplicity, the simulator will only simulate processes executing on a single-core processor, and every process will be CPU bound, meaning there will be no simulation of input and output. This simulation is different than what actually happens with modern-day operating systems because in a real environment, one can only predict what the burst will be. For the simulation, the burst will be given at run-time and will be at a set length. There will be no difference in the processes that are executing, other than the schedulers they are executing on. This simulation will execute a total of 15 processes all with different burst, priorities, arrival times and deadlines.


```
Pid     Bst     Arr     Pri     Dline   I/O
1       85      6       12      123     0
2       34      3       5       234     0
3       345     1       34      465     0
4       111     30      45      230     0
5       453     17      57      560     0
6       97      45      43      345     0
7       694     56      71      897     0
8       84      23      6       126     0
9       345     13      15      435     0
10      45      7       88      435     0
11      123     23      32      325     0
12      558     14      11      725     0
13      17      34      15      343     0
14      153     23      63      502     0
15      90      54      55      435     0
```

These processes are read into the simulator through a text file. Any process with negative numbers will be removed before the simulation takes place.

2. Multi-level feedback queue

This multi-level feedback will simulate up to five queues. The number of queues will be decided by the user. Every queue except for the last one will use round robin. The round robin queues will use a user-inputted time quantum. Processes will enter based on their arrival times and then by their given priorities. If a process cannot execute within the given time quantum, the process will be demoted to the next queue. On every subsequent queue, the time quantum will be doubled, allowing more time for continuous execution. This will reduce the number of context switching needed. Processes that cannot complete their burst in the round robin queues will be moved into the last queue, which will execute by first come, first serve and will run until completion. To compensate for processes with low priority that are not able to execute in the first come first serve queue, the process will age. Aging is when a process is not allowed execution time because of low priority. If this user-defined aging interval is met, the process will be promoted to the last round robin queue and will execute there the next time it is allotted CPU time.

3. Real time

This schedule design has only one factor deciding which process is allowed to run: its deadline.

The process with the closest deadline will run until completion or a process with a lower deadline demands CPU time. If two processes have the same deadline, the processes with the lowest priority will execute. For every clock tick that happens, a check will occur for processes that are arriving in the queue. If a new process has the lowest deadline, it is allow to execute. For real time schedulers, there are two different types: hard and soft. Both factor in if a process can execute before its given deadline. For soft real time, the process is not allowed to execute at all and is removed from the queue of processes waiting for execution. The difference with hard real time is that if one process cannot meet its deadline, the simulation will exit and no waiting processes are allowed to run.

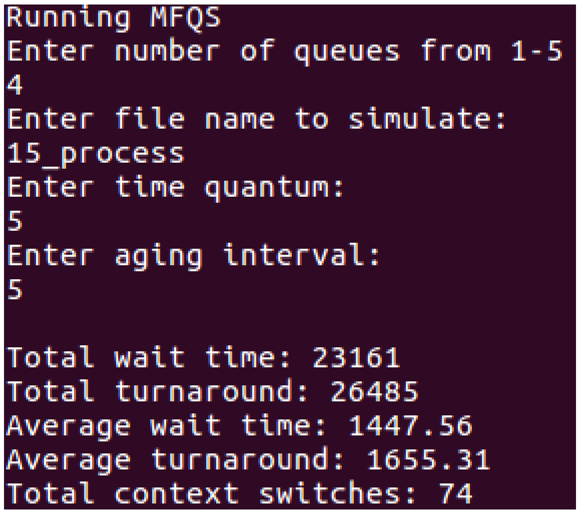
4. Simulation

As stated previously, the scheduling algorithms will be comparing the run times of 15 processes. All process attributes are randomly generated. The process deadlines have been set high in the real time scheduler. This is done to make sure every process can finish without arriving at their deadline and will lead to data that is more accurate. If a process is removed because a deadline is not met, there will be fewer processes running in the real time simulator than in the feedback queue, therefore lowering both wait times and the turnaround time. A time quantum is set at five for the first queue in the multi-level feedback scheduler. The simulation will end when all processes execute their given burst and the average wait and turnaround time will be computed.

5. Multi-level feedback queue self comparison

To compare the two different schedulers fairly, the differences between the number of queues running must also be evaluated. As the simulator returned, the wait time and turnaround times for a feedback queue running only two different queues (one round robin and the other first come, first serve) was about five seconds slower than the feedback queue running five different queues. Another thing to notice is the fact that any simulation that had three to five queues had almost identical wait and turn around times. This simulation was comprised of processes with small burst times (nothing over 50). The other thing to note is that the amount of context switching stayed constant. While this simulator doesn't account for context switching, in real world environments context switching would also add onto the total wait time of the processes. Therefore, while comparing the different amounts of queues, running context switching should not be overlooked.

What happens if processes have larger burst times? With lager burst times, processes will move into lower queues and, because of that, more context switching will occur. The amount of context switching is not the only thing that changed. The average wait times reversed, with the simulation with only two queues having the shortest wait and turnaround times. With only two queues, processes move into the first come first serve queue faster. Processes spend less time waiting because there is less context switching and process are allowed to run for longer periods of time. The simulation with only two queues had 47 context switches, in comparison to the simulation with five queues that had 76 context switches. Due to the fact that the only thing being compared between the two schedulers are the average wait and turnaround times, the different wait times and turnaround for each multi-level feedback queue will also be averaged.



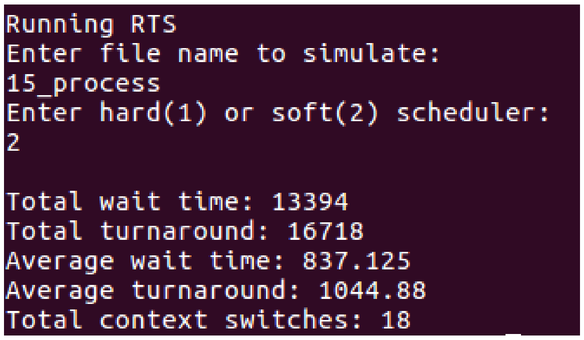
```
Running MFQS
Enter number of queues from 1-5
4
Enter file name to simulate:
15_process
Enter time quantum:
5
Enter aging interval:
5

Total wait time: 23161
Total turnaround: 26485
Average wait time: 1447.56
Average turnaround: 1655.31
Total context switches: 74
```

The average wait time for the multi-level feedback queue is 1417.86 clock ticks and the average turnaround is 1625.61 clock ticks with a time quantum of five.

6. Real time simulation

This simulation is a lot more straight forward because of the much simpler design. The only thing that matters is the deadline, and if it is the closest one, that process will execute until it finishes or another process with a low deadline arrives. It is vital to also look at the different wait and turnaround times for processes with small bursts and ones with large ones. Processes with smaller bursts had an average wait time of 91.81 clock ticks and an average turnaround of 108.63. If we increase the burst times, the average times increased.



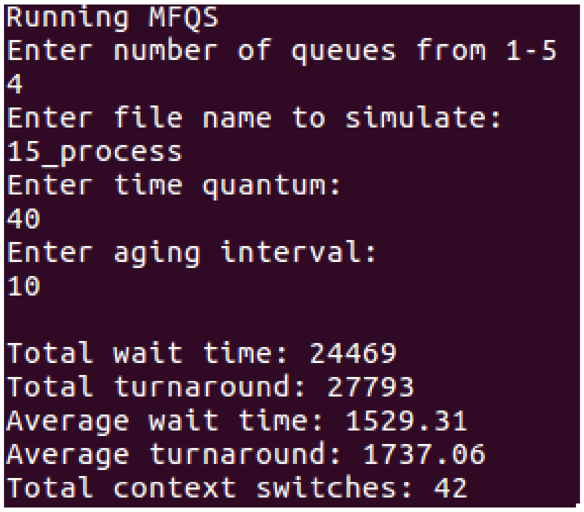
```
Running RTS
Enter file name to simulate:
15_process
Enter hard(1) or soft(2) scheduler:
2

Total wait time: 13394
Total turnaround: 16718
Average wait time: 837.125
Average turnaround: 1044.88
Total context switches: 18
```

The average wait time compared to the multi-level feedback queue is 837.13 clock ticks and an average turnaround of 1044.88 clock ticks. The number of context switches that occurred was 18.

7. Comparing the schedulers

The average times between the two schedulers is quite large with the real time scheduler completing in almost half the time as the multi-level feedback queue. This could be because the real time scheduler is a much more efficient scheduler. If the process characteristics remain static and the average times for the feedback queue only get lager with more queues, there is only one characteristic within the feedback queue that we can change. If the time quantum is increased to 40, the average times change drastically. If the time quantum was increased to 40, the average wait time increases to 1529.31 and an average turnaround of 1737.06.



```
Running MFQS
Enter number of queues from 1-5
4
Enter file name to simulate:
15_process
Enter time quantum:
40
Enter aging interval:
10

Total wait time: 24469
Total turnaround: 27793
Average wait time: 1529.31
Average turnaround: 1737.06
Total context switches: 42
```

This makes the feedback queue scheduler even more inefficient than the real time scheduler. The only characteristic that benefits from a larger time quantum is the amount of context switches. They drop from 63 with a time quantum of 5 to 50 with a time quantum of 40. Even with 23 fewer context switches, the real time scheduler is still less than double that of the feedback queue scheduler. One simulation that might change the averages is if the real time scheduler would run its worst case scenario. That would happen if the processes with the longest deadline arrive first. This would cause the scheduler to make context switches every time a new process arrived and, depending on how long these context switches take, this could allow the feedback queue to be more competitive with the real time scheduler.

8. Summary

No matter how the characteristics of the feedback queue scheduler changes, it will always be less efficient than the real time scheduler in the simulations. The real time scheduler has not only a shorter average for both wait and turnaround times, but it also does less context switching. This simulation is a small representation of everything that take place when scheduling processes and more complex simulations could render different outcomes. For this simulation, it appears that the real time scheduler is more efficient in every statistic than the multi-level feedback queue.