Data Analysis Report

Tom Dale, Alex Gonzalez, Nick Little

4/9/2018

Parallelism and percent speedup:

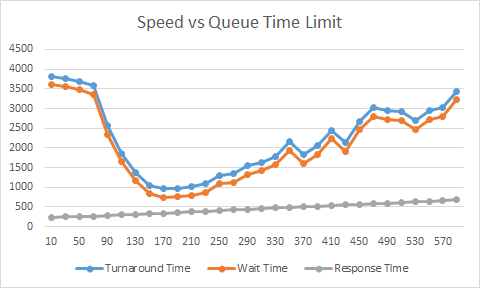
Our data analysis was done to find the optimal configuration of queue timings and processor timings. We did not implement a uni-processor simulation because this would require us to create a completely new simulation with different multilevel feedback queue design. Our model is designed for parallel processing so this wouldn’t be an accurate metric. We can use the knowledge that for 6 processors and a 75 % parallelism, found by our average process being calculated 4 threads, the theoretical speedup is .

TEST 1: Speed vs. Queue Time Limit

Info: This test is being conducted by changing the amount of time each process can spend in a queue STR, RR1, RR3, RR5, or RR10. The final queue is FCFS, processes that reach it will stay until completion. The test takes a static amount of time associated with the first five queues and multiplies it by a “timeLimitFactor”. After all the processes are completed, the timeLimitFactor is then incremented by 20 time units (starting at value 10) for 30 runs. The purpose of this test was to see at what time multiplier our system had the best turnaround, wait, and response averages. Round Robin Time factor was kept at 10, and context switching was kept at 5.

Assumption: With a shorter amount of time being spent in each queue, more processes would be responded to, but longer processes would pay the price. Shorter processes would be favored in this case, and the longer processes would be spending more time in our final queue. If the amount of time spent in each queue was too much, the shorter processes would pay the price.

Results:



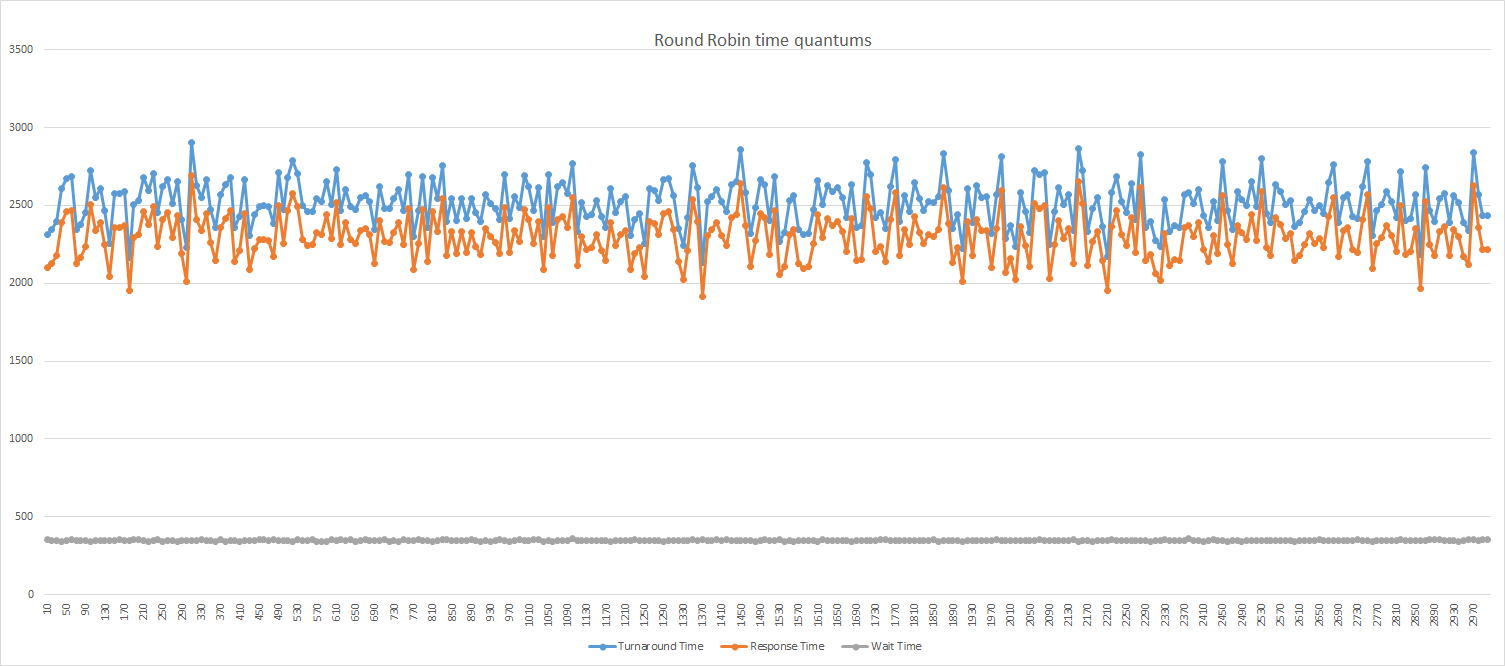
Our results were exactly what we expected. The most optimal time multiplier was 170t. This is where the average wait time and average turnaround time were the shortest. While the average response time was smallest when processes spent less time in each queue, more processes were spending less time waiting and had faster completion. After 170t, the averages began to rise.

TEST 2: Round Robin Time Quantums

Info: For this test we kept the amount of time spent in each queue at the optimal 170t, context switching was kept at 20t, but we changed the time slicing values for each of our round robin queues. We used a similar method for the test above. We changed the values by multiplying a static value associated with each RR by a “RRtimeFactor”. The RRtimeFactor was incremented by 10 after our system was completed. We tested this for 300 different values. The purpose was to see how longer time slices would help larger processes complete faster, which would lower the turnaround average.

Assumption: By increasing the amount of time slicing in each RR, there would be a better turnaround rate, but it may punish smaller processes. The wait time may be much longer because of longer processes taking their full time slices to complete.

Results:



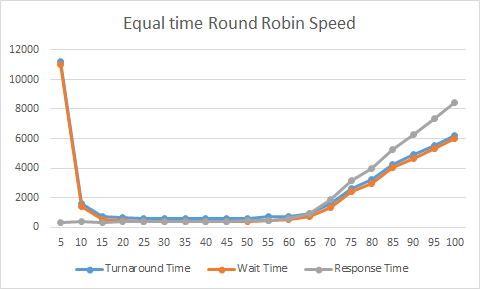
Our assumptions weren’t terribly wrong, but there is no way to draw any concluding evidence as there are spikes all along the graph. What we could assume from this is that there are certain thresholds which would favor the data set we used. Ultimately without a massive context switch time, the round robin time quantum choices wont dramatically speed up our simulation.

TEST 3: Equal Time Round Robin

Info: After the previous round robin test was completed, we experimented with rearranging the order of processes queues in our multilevel feedback design. The optimal order of our processors was actual the standard order that we started with, while the worst order actually the reverse: FCFS-> RR10->RR5->RR3->RR1->STR. We also tried making the round robin processor time quantums different ratios and found that the greatest speed came from having all four RR queues have the same time quantum.To test this we had a program that tested time quantum values from 5-100 but with all RR queues having the same value.

Assumption: The order of round robin queues was seen as somewhat arbitrary because context switching was low and all processes seemed to go far through the feedback queue regardless of earlier tests.

Results:



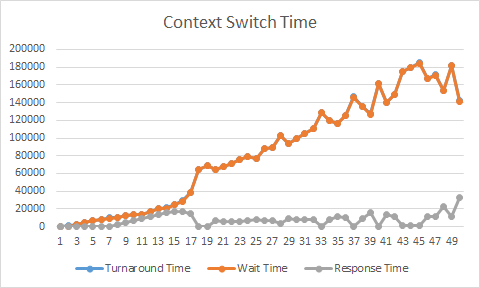
Our fastest time was achieved using this test. With a turnaround time of 587 and a wait time of 372, having 3 round robin queues with time quantum 40 was the optimal run. The graph shows how smaller time quantums run into problems with too much context switching, and larger time quantums have problems with response time.

TEST 4: Context Switch Times

Info: This test uses a value of 10 for time slicing in the round robins and 170 for queue time. We changed the value of our context switching by incrementing it after each time our system completed all processes. The purpose of this was to see context switching would make our system less optimal.

Assumption: Larger amounts of context switching would make the response time, wait time, and turnaround time rise. Our context switching values are static and don’t vary from process to process. So we should see a consistent rise on the graph.

Results:



While our assumptions were correct, we began to see some spiking in the later part of the graph. This could have been due to some of the specifics of the data set we tested on. For the most part there was a constant rise, and obviously smaller context switching resulted in better turnaround, wait, and response time.

OVERALL:

The most optimal times for our system was setting the timeLimitFactor to 170 time units. This allowed the most processes to be completed in a shorter amount of time. Also the processes didn’t have to wait as long to continue executing. While response times weren’t the most optimal, the benefits are worth it for our system. The conclusion we might draw from our second test is that we would take one of the RRtimeFactor values with the best averages, but deciding which one is truly more optimal would be difficult because there are many values that produce similar results. Moving on from test 3 resolves the problem with no optimal RRtimeFactor by showing that having all round robin queues have a time quantum of 40 is optimal. Another factor that would make it difficult to choose is differing sets of data for processes. Longer values for time slicing may benefit data sets with larger processes than with small ones. Finally, our third test suggest that having the smallest amount of context switching possible is most optimal for our system.