Louis Rozencwajg-Hays, Nicholas Gattuso, Kexian Wu

I pledge my honor that I have abided by the Stevens Honor System.

Experimentation and analysis of results

**Case 1:**

**Initial scenario**

**Algorithm: FCFS**

**./assign1 5 5 100 10 0 0 5**

**See Column B in excel sheet.**

In this first basic case, it took 2 seconds to run, with the average wait time of 1ms. Also, the throughput for both consumers and producers was around 2740. Additionally, the turnaround was around 2.

**Case 2:**

**Initial scenario**

**Algorithm: RR**

**./assign1 5 5 100 10 1 10 5**

**See Column C in excel sheet.**

In this first basic case, for round robin, it took 5x as long compared to FCFS, with the average wait time being 12021ms! Also, the through-put for both cons, and prod was around 60. Additionally, the turnaround avg was 12022.

**Case 3:**

**In this case, we have 10x the producers as consumers.**

**Algorithm: FCFS**

**./assign1 50 5 100 5 0 0 5**

**See Column D in excel sheet.**

In this next case, with more producers, the results were that the runtime was cut in half compared to the first case. The total time took 1018ms, with the avg wait time of 5ms however. This could be because of the low queue size for the amount of producers.

**Case 4:**

**In this case, we have 10x the producers as consumers.**

**Algorithm: RR**

**./assign1 50 5 100 5 1 10 5**

**See Column E in excel sheet.**

In this case, for red robin, 50 producers and 5 consumers for 100 products too 51594 ms, with an average wait time of 2276ms. This is signifigantly slower than its FCFS counterpart in case 3.

**Case 5:**

**In this case, we have more consumers than producers**

**Algorithm: FCFS**

**./assign1 10 50 100 5 0 0 5**

**See Column F in excel sheet.**

For this case, we have more consumers than producers, and the runtime was 1016ms, with the average wait time being 1ms. Also, the throughput for both consumers and producers was signifigantly higher than its previous cases, being around 5910.

**Case 6:**

**In this case, we have more consumers than producers**

**Algorithm: RR**

**./assign1 10 50 100 5 1 10 5**

**See Column G in excel sheet.**

For this case, we have more consumers than producers. The runtime for RR was 10353 ms, so signifigantly slower than its FCFS counterpart. It’s average wait time was 615ms as well.

**Case 7:**

**In this case, we have the same number of producers and consumers, but we have a large amount of products, and a slightly larger queue. The quantum has also increased to 100.**

**Algorithm: FCFS**

**./assign1 5 5 3000 10 0 0 50**

**See Column H in excel sheet.**

For this case, we have the same, small amount of producers and consumers. Also, the amount of products to be produced is very large at 3000, so naturally the runtime will be longer. The runtime is 60287ms, and its average wait time is still 1ms.

**Case 8:**

**In this case, we have the same number of producers and consumers, but we have a large amount of products, and a slightly larger queue. The quantum has also increased to 100.**

**Algorithm: RR**

**./assign1 5 5 3000 10 1 100 50**

**See Column I in excel sheet.**

For this case with RR, the runtime was 347098ms, which is very long. The average wait time was around a second, being 967ms. The throughput was also around 520 for both consumers and producers.

**Case 9:**

**In this case, we have an unlimited queue size.**

**Algorithm: FCFS**

**./assign1 5 5 100 0 0 0 50**

**See Column J in excel sheet.**

Despite the unlimited queue size, the results stay mostly similar to past similar test cases. The run time was 2022ms, with the avg wait time being 1ms.

**Case 10:**

**In this case, we have an unlimited queue size.**

**Algorithm: RR**

**./assign1 5 5 100 0 1 100 50**

**See Column K in excel sheet.**

With unlimited queue size, the results also stay mostly similar to past similar test cases, with its run time being 10829ms and its avergae wait time being 3710ms.

**Case 11:**

**Same number of producers and consumers**

**Algorithm: FCFS**

**./assign1 4 4 100 10 0 0 123**

**See Column L in excel sheet.**

For this case, the results also stay similar to past similar test cases, with its run time being 2601ms, and its average wait time being ~0ms.

**Case 12:**

**Same number of producers and consumers (very large quantum)**

**Algorithm: RR**

**./assign1 4 4 100 10 1 1024 123**

**See column M in the excel sheet.**

When the quantum approaches the maximum possible life, the long quantum degenerates into FCFS. The consumer throughput is approximately equal to producer throughput, but consumer throughput is bounded by producer throughput. This is why the results are almost exactly the same as the FCFS test case.

**Case 13:**

**Lots of producers**

**Algorithm: FCFS**

**./assign1 2000 4 100 10 0 0 123**

**See column N in the excel sheet**

For this case, with the extreme amount of producers and minimal consumers, the runtime was 5421 ms, and the average wait time was 209ms.

**Case 14:**

**Lots of producers && small quantum**

**Algorithm: RR**

**./assign1 2000 4 100 10 1 8 123**

**See column O in the excel sheet**

For this red robin case, the run time was long once againm running in at 178669ms, and average wait time being 21747ms! Also, the throughput was extremely long at 35 for both consumers and producers.

**Case 15:**

**Lots of consumers**

**Algorithm: FCFS**

**./assign1 4 2000 100 10 0 0 123**

**See column P in the excel sheet**

For this case, the run time was 3409ms, and average run time being 2ms. This is because of the high demand of product from the massive amount of consumers, and minimal producers.

**Case 16:**

**Lots consumer**

**Algorithm: RR**

**./assign1 4 2000 100 10 1 1024 123**

**See column Q in the excel sheet**

By comparing case 15 and case 16, we observed that the RR algorithm has a much faster wait time then FCFS algorithm when there are a large number of consumers. It also has a very similar runtime, at 3404ms.

**Case 17:**

**Lots of Products**

**Algorithm: FCFS**

**./assign1 4 4 1000 10 0 0 123**

**See column R in the excel sheet**

For this straight forward case, the runtime was 25805ms, and the average wait time was 1ms. The long wait time is obviously caused by the larger amount of products and minimal consumers and producers.

**Case 18:**

**Lots of Products**

**Algorithm: RR**

**./assign1 4 4 1000 10 0 1024 123**

**See column S in the excel sheet**

By comparing case 4 and case 9, we observed that FCFS is faster than RR for both average turnaround time and average wait time when the number of products is large and quantum time is equal to average life time.

**Case 19:**

**Small number of Products**

**Algorithm: FCFS**

**./assign1 4 4 50 10 0 0 123**

**See column T in the excel sheet**

For this case, the runtime was 1361ms, and the average wait time was 1ms.

**Case 20:**

**Small number of Products**

**Algorithm: RR**

**./assign1 4 4 50 10 0 1024 123**

**See column U in the excel sheet**

For this case, once again we have very similar numbers to the FCFS counterpart. The runtime was 1347ms, with the average wait time being ~0ms. This is due to the high quantum.