Carlo Quitorio

CSCI 114 Bounded Buffer Queue Project Report

Introduction

This project involved the implementation of a Bounded Buffer Queue (BBQ), as shown in the textbook on figure 5.8, with a total of 20 threads where 10 were producers and 10 were consumers.

Hardware & Software used

A desktop was used running Visual Studio Code along with cygwin64 to do the compiling and execution of the program.

Implementation

The delay of the producing (TP) and consuming (TC) threads in ms, along with the size of the queue (QS) are passed into the program. Producers would create numbers at random times, ranging from 0 to TP, and then insert them into the queue and consumers would pop those numbers out of the queue. The producers would speed up (approaching 2 times speed) as the queue got emptier and gradually slow down or stop as the queue passed 75% total capacity. Consumers would pop the numbers from the queue at a random time in ms ranging from 0 to TC.

The instructions desired that the program would run indefinitely, but for recording data the program would run for a total of 5 seconds. After which the number of items produced, consumed, the number of stopped producers, and consumers would be output by the program and be recorded. The delays for TP and TC, and the size of the queue are recorded before the execution and can be seen in the command to run the program.

\

Data

Producer Halt rate is = Producer Halts / #itemProduced

Consumer Halt rate is = Consumer Halts / #itemConsumed

Producer						Consumer			
Queue Size	Sleep Range(ms)	Halts	#itemProduced	Halt Rate in %		Queue Size	Sleep Range (ms)	Halts	Halt Rate in %
25	10	25	125	20.00		25	10	5	4.00
25	20	9	65	13.85		25	20	6	9.23
25	80	5	50	10.00		25	80	6	12.00
25	200	5	40	12.50		25	200	5	12.50
25	500	5	30	16.67		25	500	5	16.67
25	1000	5	40	12.50		25	1000	0	0.00
50	10	5	265	1.89		50	10	5	1.89
50	20	5	125	4.00		50	20	6	4.80
50	80	5	90	5.56		50	80	6	6.67
50	200	5	125	4.00		50	200	0	0.00
50	500	6	80	7.50		50	500	0	0.00
50	1000	0	45	0.00		50	1000	0	0.00
100	10	37	555	6.67		100	10	13	2.34
100	20	6	230	2.61		100	20	5	2.17
100	80	10	225	4.44		100	80	6	2.67
100	200	0	200	0.00		100	200	0	0.00
100	500	0	95	0.00		100	500	0	0.00
100	1000	0	45	0.00		100	1000	0	0.00

Note: the items produced matched the items consumed at every execution of the program, thus it was not recorded to make the table more concise.

Graphs are included in the .zip file for viewing.

Discussion of the Data

At QS = 25 as the delay range of the threads increased, fewer items were produced and consumed, along with this the number of halts also initially decreased. This is due to the small queue size, at lower delay ranges the producers were able to outpace the consumers and fill up the queue causing the producers to stop. As the delay ranges increased the consumers were not able to keep up with the producers and resulted in an increase in their halt rate.

Much like the QS = 25, QS = 50 has a decreasing number of items produced as the delay ranges increase for the threads. The producers had a consistent amount of halts and resulted in an increase in the producing halt rate over time. Consumers however were able to reach no halts at higher delay ranges, probably due to the much higher delay range producers get as the queue fills up, and resulted in a 0% consumer halt rate.

For QS = 100, at lower delay ranges both consumers and producers had higher halt rates. The producing thread was able to increase speed quickly and filled up the queue far faster than the consumer could empty it. As the delay ranges increased the producers weren't able to increase their speed as fast and both consumers and producers had a 0% halt rate.

Conclusion

At lower queue sizes the halt rates of the producers and consumers increase due to the queue filling up much faster and causing the producers to wait for the consumers. As the queue sizes increase the halt rate decreases due to there being more space for producers to fill up and consumers having more time to empty the queue. Larger queue sizes also allow for the production and consumption of larger quantities of data, as shown by the tables. Smaller delay ranges also allow for the production and consumption of data to increase drastically, but at the cost of an increase in the amount of halts. Ideally a system should balance the production and consumption carefully to achieve maximum input and output at smaller delays without causing a large amount of halts.

Problems Encountered

Due to the consumer and producer threads having differing delay ranges, outputting which item was created by which producer and consumed by which consumer proved to be very difficult, but doable. However doing this takes away from the concept being taught so the output of the program is far from readable, in order to make it readable the ranges of the producer and consumer delay ranges had to be drastically increased, sometimes by 2000 ms, that it would prevent the program from fully displaying what threading can achieve. The code will be submitted without the proper changes so output will be hard to read if the correct delay ranges are not passed through.