Analysis of Average Waiting Time in a Producer-Consumer System

Rousomanis Georgios (10703)

May 2025

1 Introduction

The producer-consumer problem is a classic example of a multi-thread synchronization issue, where multiple producer threads generate data and insert it into a shared queue, and multiple consumer threads remove and process this data. In this report, we analyze the behavior of such a system, focusing on the average waiting time of items in the queue, defined as the duration between the time an item is enqueued by a producer and the time it is dequeued by a consumer.

2 Experimental Setup

The experiment was performed on a 4-core machine using a custom implementation of the producer-consumer pattern written in C, utilizing POSIX threads (pthreads). The system includes:

- A fixed-size FIFO queue of size 10.
- Each producer performs 10,000 iterations of adding dummy work to the queue.
- Consumers retrieve and immediately process this work.
- No artificial delays (e.g., `sleep()` or `usleep()`) are introduced in either producers or consumers.
- The average waiting time is measured for each item from the moment it is enqueued until it is dequeued.

The main variable parameters are:

- $P \in \{1, 2, 4, 8\}$: the number of producer threads.
- $Q \in [1, 20]$: the number of consumer threads.

3 Results and Discussion

Figure 1 presents the average waiting time (in microseconds) as a function of the number of consumer threads Q, for different values of producer threads P.

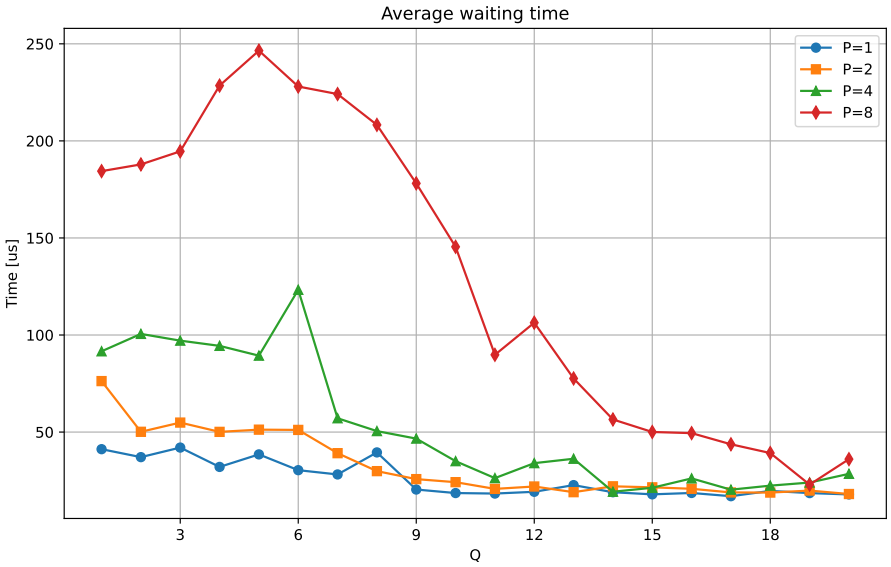

Figure 1: Average waiting time vs. number of consumer threads Q, for various values of P.

The observed trends can be summarized as follows:

- For a small number of consumer threads, the queue tends to fill up quickly, causing producer threads to block and resulting in higher waiting times.
- As Q increases, consumers are able to process items more promptly, decreasing the average waiting time.
- When Q becomes sufficiently large, the waiting time decreases sharply and then tends to plateau, indicating that adding more consumers yields diminishing returns.
- Higher values of P consistently result in higher waiting times for the same Q, due to increased contention for the shared queue.

Notably, for $P = 8$, the system becomes significantly more congested, especially when Q is low. This reflects a situation where production outpaces consumption, causing the queue to reach capacity more frequently and introducing delays for both producers and consumers.

4 Conclusion

The experiment highlights the importance of balancing producer and consumer threads in a multi-threaded system. Under-provisioned consumer capacity leads to longer waiting times and reduced system responsiveness. On a 4-core system, optimal performance is generally achieved when the number of threads is aligned with the hardware's parallel capabilities. For compute-light workloads such as this one (dummy work), consumer saturation occurs quickly, and additional threads do not substantially improve throughput.

These insights are essential for designing concurrent applications that require efficient inter-thread communication, especially under real-time constraints.

- Producer-consumer problem source code:
  `https://github.com/georrous6/Real-Time-Embedded-Systems.git`
- K-nearest neighbors source code:
  `https://github.com/georrous6/Parallel-and-Distributed-Systems.git`