Producer-Consumer System Analysis

Ioannis Michalainas

April 2025

Abstract

This assignment investigates the performance of a multi-threaded producer-consumer system where p producer threads generate computational tasks and q consumer threads process them. The average task waiting time is measured to evaluate the impact of consumer count on performance and to identify optimal thread configurations.

1 Introduction

The implemented system uses a shared FIFO queue to coordinate work between producer and consumer threads. Producers generate computational tasks involving sine evaluations of the form $\sin(0.1k\pi)$ for $k = 1, \ldots, 10$, which are then processed by consumer threads. Key design elements include:

- Bounded queue capacity: 10 tasks
- Thread synchronization via mutexes and condition variables
- Poison pill termination mechanism

2 Methodology

Each producer thread generated 100,000 tasks. Timestamps were recorded at the moment of task enqueueing using `gettimeofday()` and consumers measured the elapsed time upon dequeuing to determine each task's residence time in the queue. For each producer count (1–5), the number of consumer threads was varied from 1 to 8. The results were averaged over five runs per configuration to minimize variability. Visualization was done using Python's `matplotlib`.

3 Results and Discussion

3.1 System Specifications

The experiments were conducted on an Arch Linux system with the following hardware and system specifications:

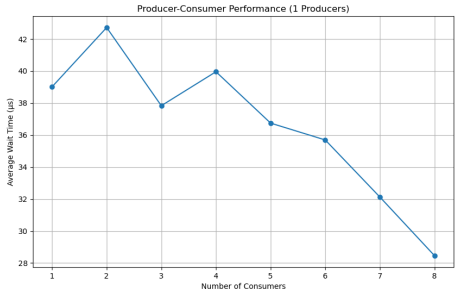
- **Processor:** Intel(R) Pentium(R) Silver N5030 CPU [1.1 GHz - 3.1 GHz]
- **Cores:** 4 physical cores (no hyper-threading)
- **Memory:** 4 GB RAM

3.2 Waiting Time Trends

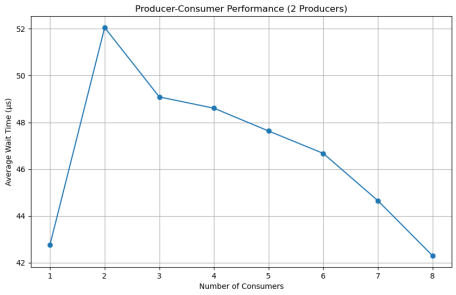
Figure 1 shows the average task waiting time as the number of consumer threads increases for different producer counts. Overall, increasing consumer threads tends to reduce task waiting time, although the effect varies depending on producer load.

For 1 producer (Figure 1a), the average waiting time decreased from 39.01 μs (1 consumer) to 28.46 μs (8 consumers), indicating significant performance improvement with more consumers. The trend continues for higher producer counts. For instance:

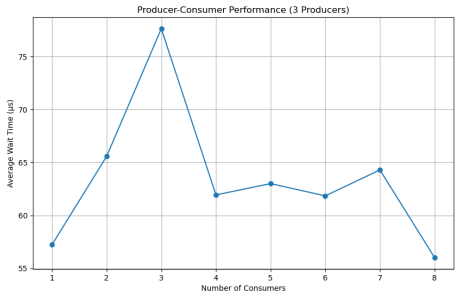
- With 2 producers, wait time decreased from 42.77 μs to 42.30 μs.



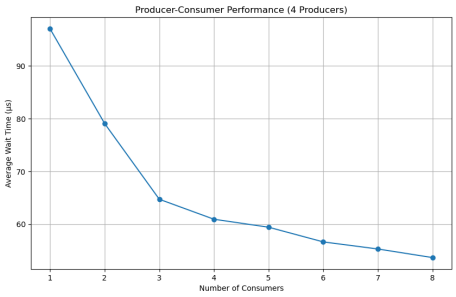
(a) 1 Producer



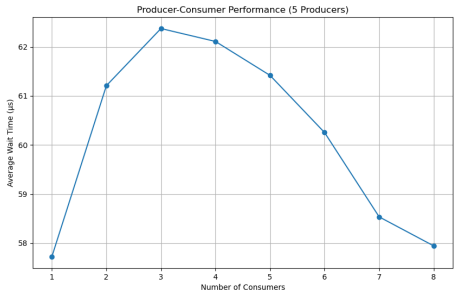
(b) 2 Producers



(c) 3 Producers



(d) 4 Producers



(e) 5 Producers

Figure 1: Average task waiting time versus number of consumer threads across producer configurations.

3.3 Explanation of Trends

The observed behavior can be attributed to three main factors:

1. **Trivial Task Complexity:** Each sine computation requires approximately 0.1 μs—significantly less than typical thread scheduling overheads (1–10 μs). This results in consumers remaining idle and immediately ready for tasks, minimizing queue delays.

2. **Queue Dynamics:** The queue has a limited capacity of 10 tasks. When full, producers block until a slot becomes available. Increasing the number of consumers helps drain the queue more rapidly, reducing both residence time and producer blocking.

3. **Efficient Thread Scheduling:** Although the N5030 lacks hyper-threading, its low context-switch latency and efficient scheduling allow lightweight consumer threads to execute concurrently with minimal overhead.