Operating Systems and Distributed Systems
Project 1 Part 1

Zhen Xuan

November 7, 2023

1 Implementation

To work concurrently, we use one enqueue thread to push image names. When finished, it close the channel.
We set up several dequeue threads to resize images. Each dequeue thread constantly gets an image and resize. When dequeue returns error code, channel is empty and thread will stop.
When measuring latency, we support multiple enqueue threads and dequeue threads. When a thread performs an enqueue or dequeue, we measure its time and sends to another channel. Main thread will use this channel and collect timing result. Since totally 10000 enqueue and dequeue, main thread will also collect each operation 10000 times and then return. Thus no dead lock will occur.

2 Result

We test our bounded queue's throughput and latency with different settings of thread number and capacity.

2.1 throughput

Our result is as follows: Fig. 1
Here baseline is sequential resizing's throughput. We use one enqueue thread and multiple dequeue threads. We notice that the throughput gets larger as thread number grows. But not change very much with more capacity. This shows that capacity is not the bottleneck. More threads brings more concurrency, which will certainly increase the throughput.

2.2 latency

We measure enqueue and dequeue latency separately. Specifically, we test different ratio of enqueue thread number and dequeue thread number. Our result is as follows: Fig. 2, Fig. 3 The reason we measure latency separately and with different ratio is that we find dequeue latency is significantly larger than enqueue latency. (Enqueue latency is usually several microseconds, while dequeue is several thousands.)
This suggests that enqueue is not the bottleneck as well as capacity, which also confirms what we find in throughput.

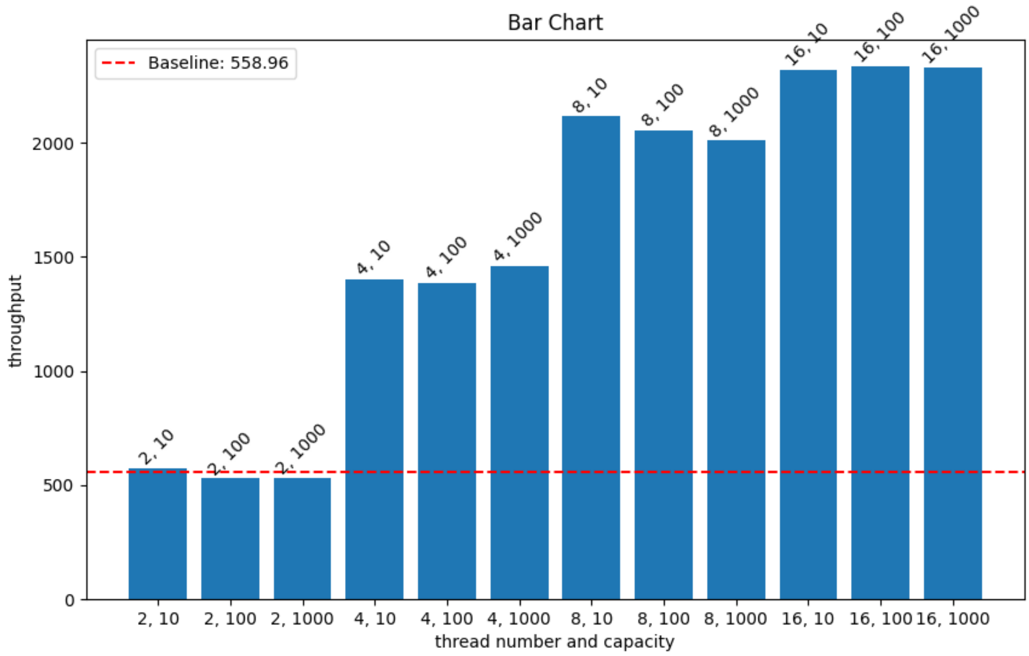

Figure 1: throughput with different (thread numbers, capacity)

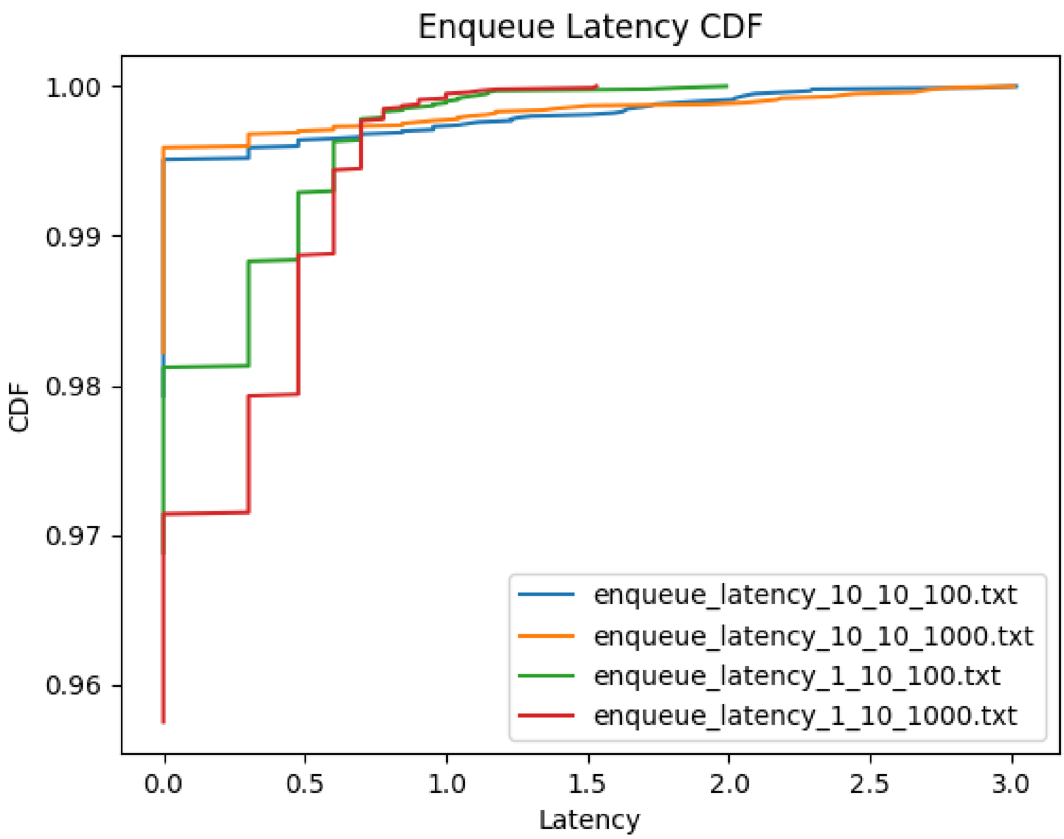

Figure 2: lantency with different (enqueue, dequeue, capacity)

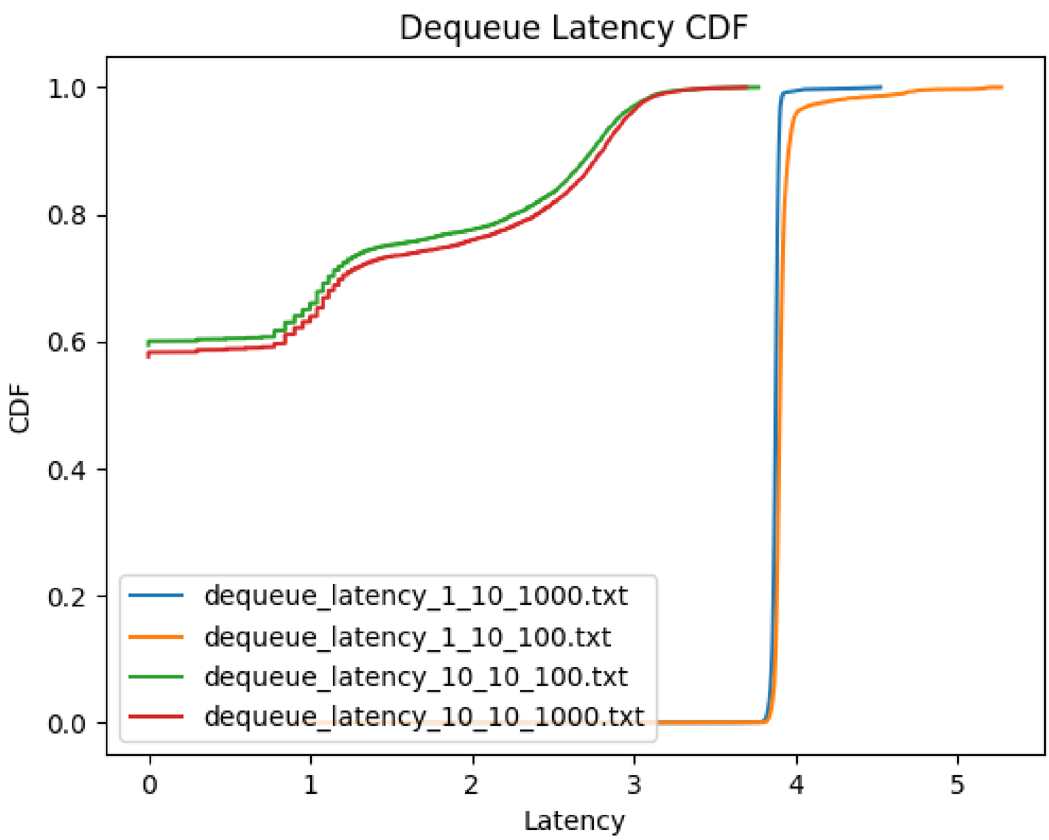

Figure 3: lantency with different (enqueue, dequeue, capacity)

With larger ratio of enqueue over dequeue, we see that dequeue latency drops, which strongly proves that enqueue is the bottleneck.