Task 1: Measure Unloaded Latency

Implemented a closed-loop client that sends 1000 synchronous proposals to a 3-replica Raft cluster. Measured latencies to compute average (latAvg), median (latP50), and 99th percentile (latP99) latency.

Here N=1000, since all the 1000 synchronous proposals are committed.

Calculating latAvg

Calculated using the formula: $latAvg = \frac{\sum_{i=1}^{N} \text{latency}_i}{N}$

Calculating latP50 and latP99

Sorted latencies and computed indices: $\text{Index}_p = \left\lceil \frac{p}{100} \times N \right\rceil - 1$

Retrieved latency at Index_{50} for latP50 and at Index_{99} for latP99.

```
#######################################
#      latAvg       latP50      latP99
#                     (ms)        (ms)
---------------------------------------
       30.297       28.566       50.881
```

Task 2: Measure Maximum Throughput of System

Developed a multi-threaded closed-loop clients script to measure the maximum throughput of the Raft system. It runs multiple rounds with an increasing number of concurrent clients (threads), each sending 1000 synchronous proposals. Here, we are doubling the number of clients after each round.

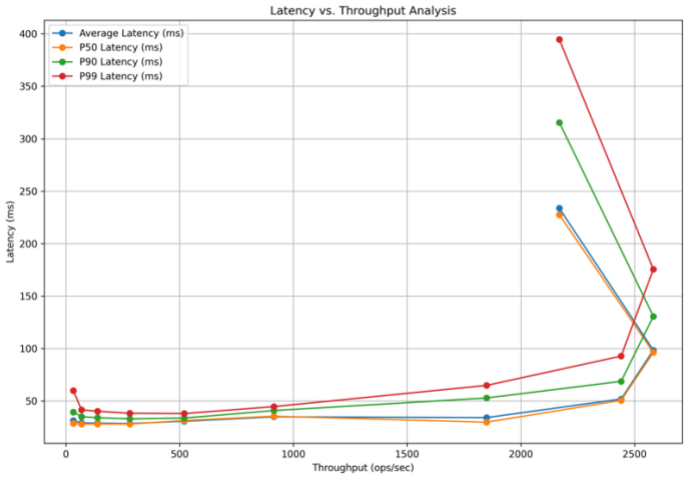
- **Closed-Loop Clients**: Each client thread sends 1000 synchronous proposals using `ctrl.propose_to_all_sync()` and measures the time taken for all proposals.
- **Total Operations**: Summed the successful proposals from all client threads.
- **Throughput Calculation**:
    - Total time is measured from the start to the end of all client threads.

$$\text{Throughput (ops/sec)} = \frac{\text{Total Successful Operations}}{\text{Total Time (seconds)}}$$

```
#############################################################################
# clientCount      latAvg       latP50       latP90       latP99        throughput
#                    (ms)         (ms)         (ms)         (ms)         (ops/sec)
-----------------------------------------------------------------------------
          1       31.240       28.625       39.496       60.022          32.008
          2       29.138       28.064       35.023       41.620          68.635
          4       28.887       28.111       34.115       40.245         138.466
          8       28.555       28.064       33.023       38.332         280.132
         16       30.715       31.435       33.799       38.051         520.417
         32       34.938       35.505       40.917       44.677         914.545
         64       34.171       29.833       52.896       64.876        1849.775
        128       51.934       50.621       68.743       92.772        2441.231
        256       98.384       96.274      130.692      175.579        2582.946
        512      233.701      227.544      315.485      394.577        2169.425
```

Task 3: Measure and Draw Latency-Throughput Plot

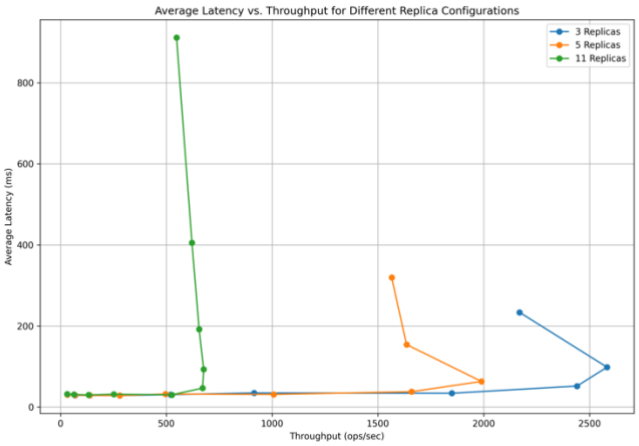
We plotted the latency and throughput measured from Task 2 to visualize the performance characteristics of the Raft system under varying client loads.

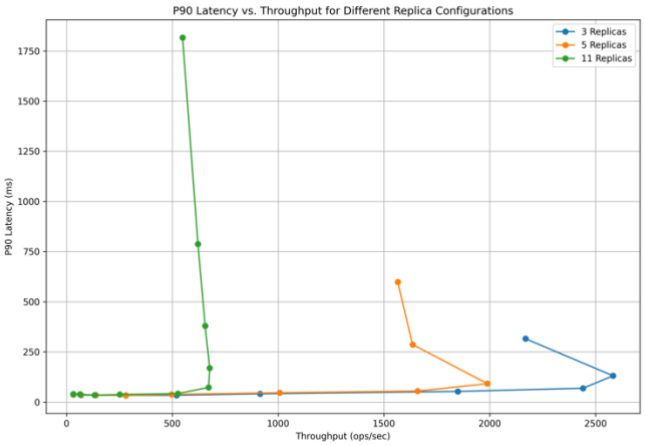


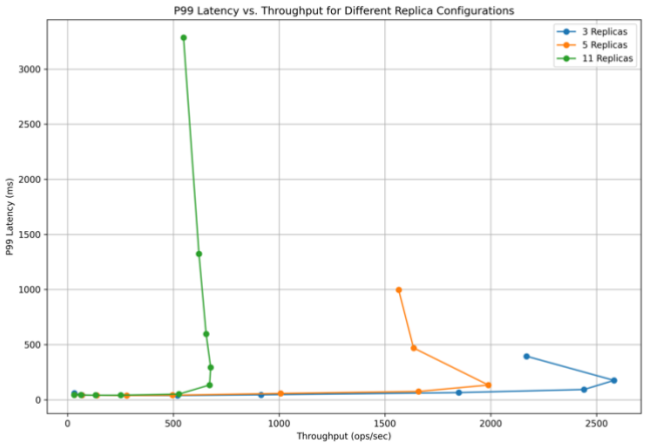
The latency-throughput plot illustrates that the Raft system can handle increasing client loads effectively up to a certain threshold (around 256 clients). Beyond this point, the system experiences increased latencies and reduced throughput due to resource saturation and overheads associated with managing more concurrent operations. This analysis helps identify the optimal operating range for the system and highlights the importance of balancing client load to maintain performance.

Task 4: Performance After Configuration Change

Re-measured the throughput and latency after increasing the number of replicas in the Raft cluster to 5 and 11. The goal was to observe how the replication factor affects system performance.







Explanation of Performance Changes

As the number of replicas increases, the system's average latency increases, and the maximum achievable throughput decreases. This performance degradation is due to the additional communication overhead and synchronization required among more replicas in the Raft consensus algorithm. With more replicas, each proposal must be replicated to a larger number of nodes before being considered committed, increasing the time taken per operation. Consequently, the system processes fewer operations per second, reducing throughput.