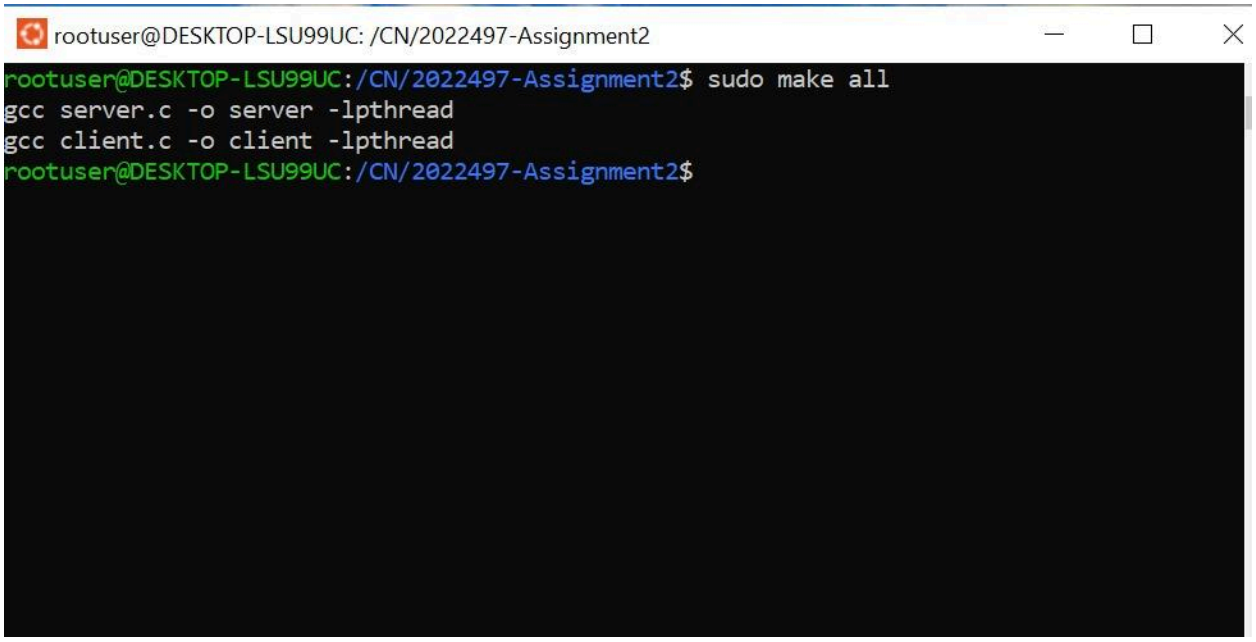
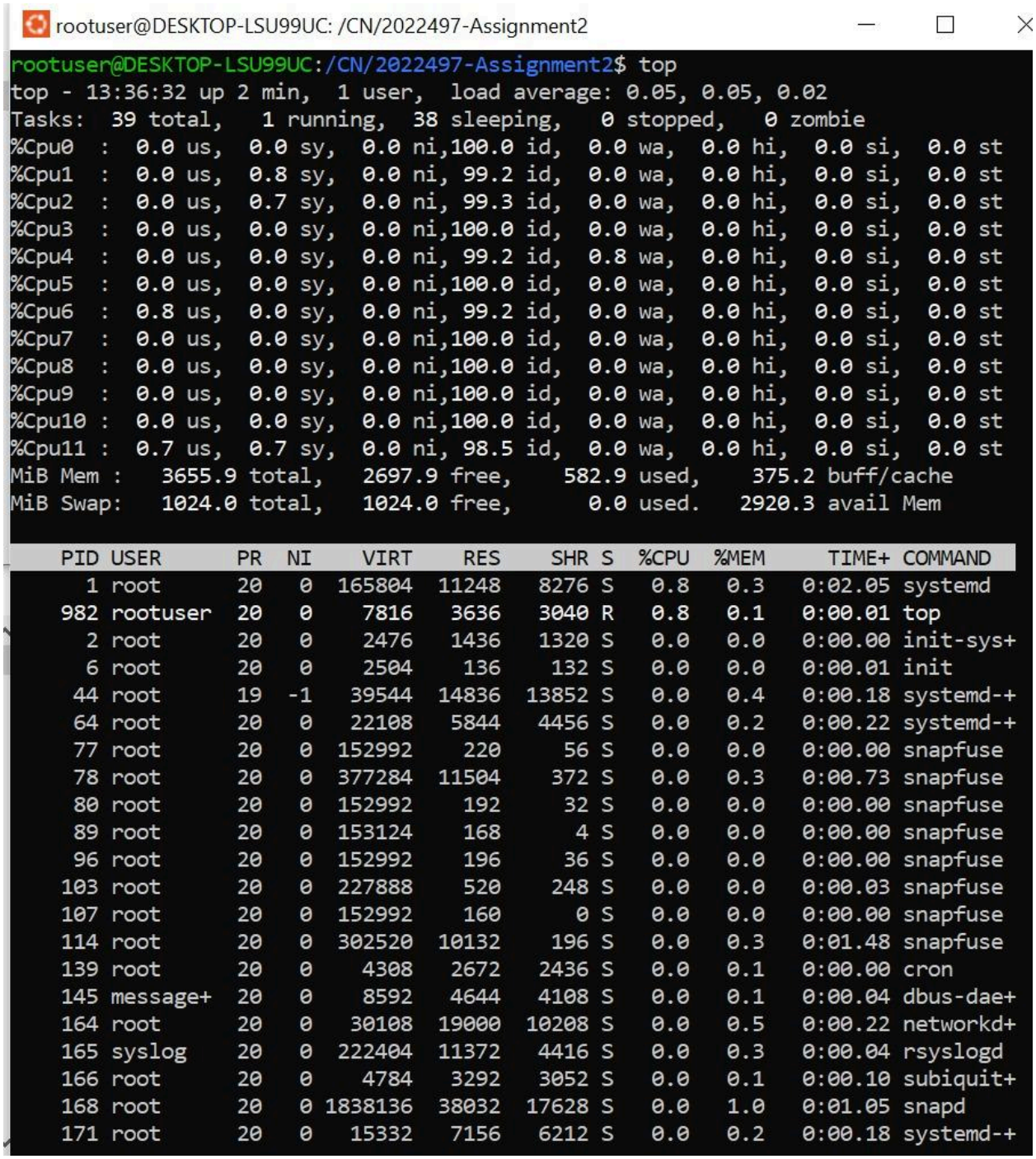
Report

Question1

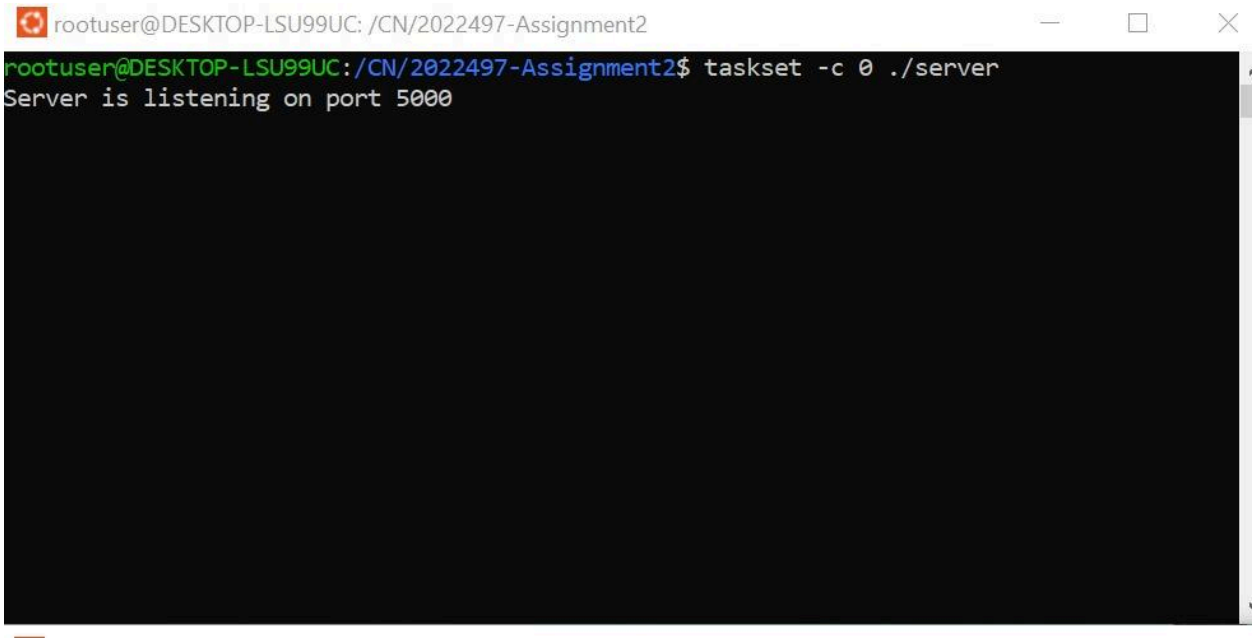
Compile server.c and client.c into executable file



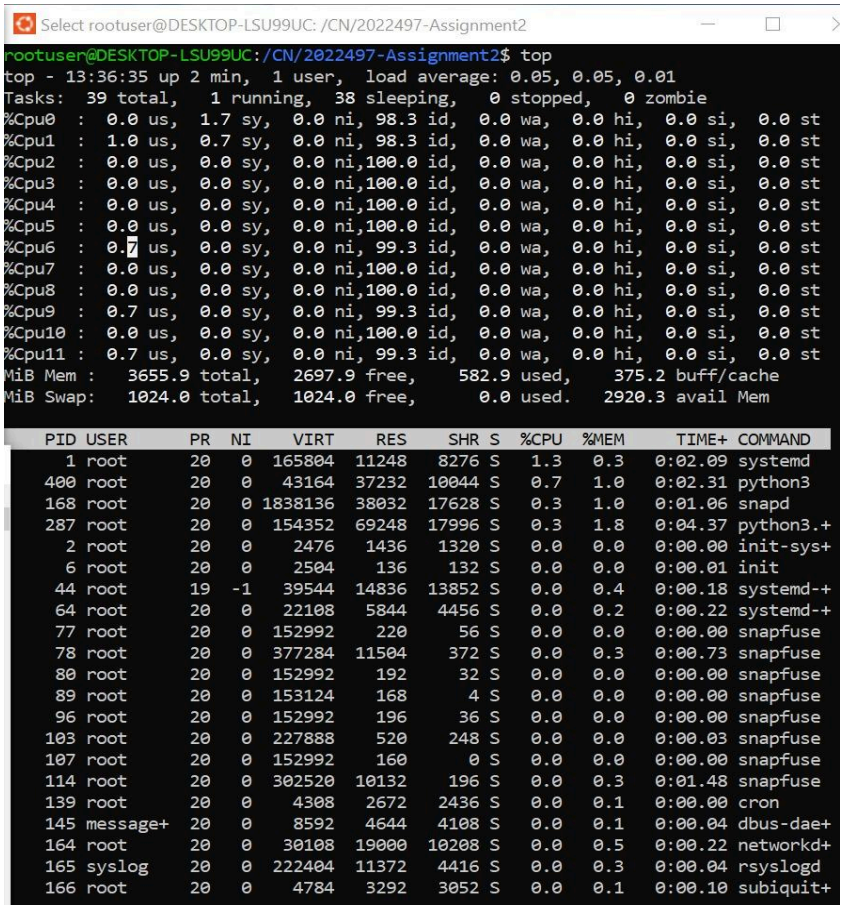
Initial configuration of the CPUs: using top command



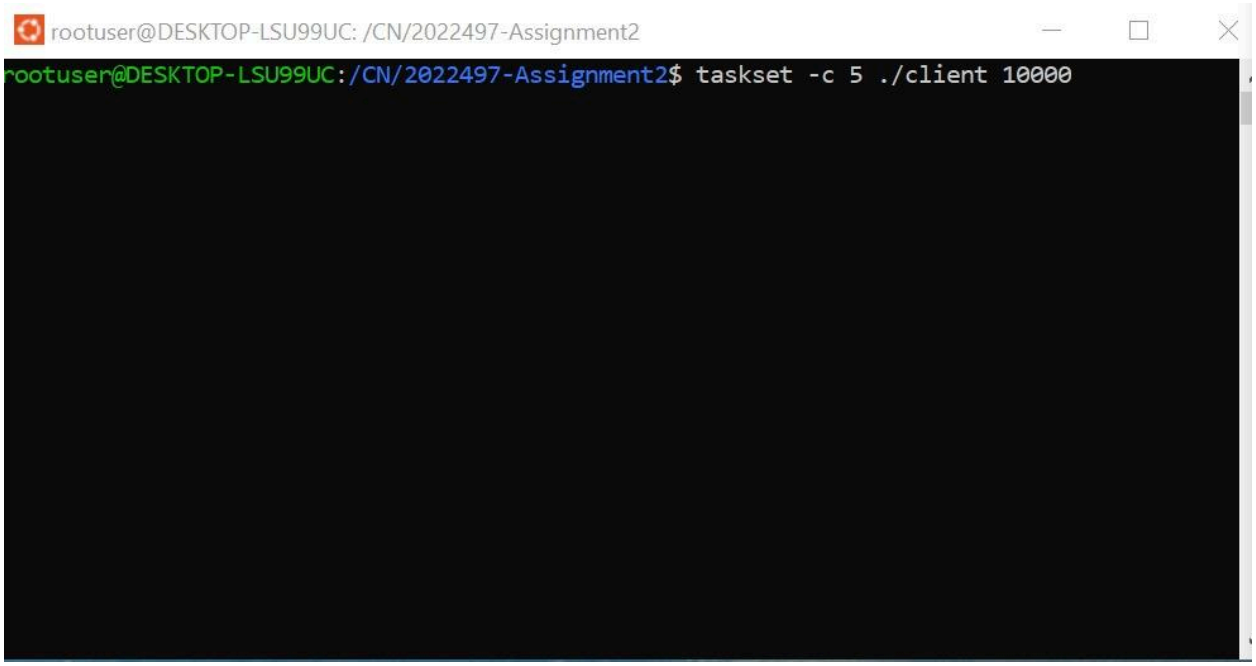
Execute the server using taskset and pin it to CPU 0:



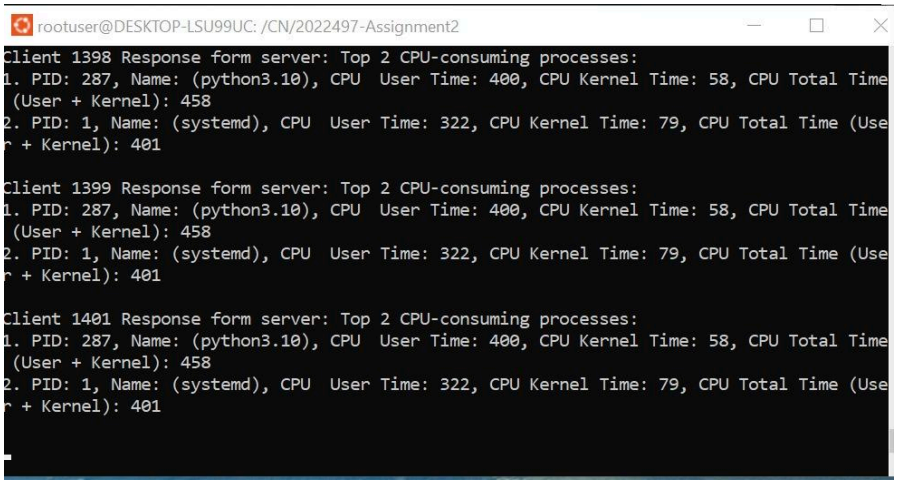
Configuration of CPUs after starting server:

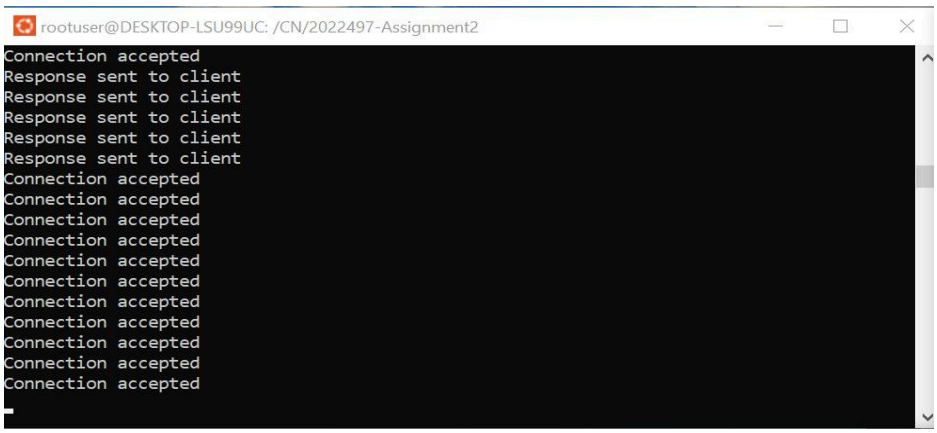


Execute the client using taskset and pin it to CPU 5:

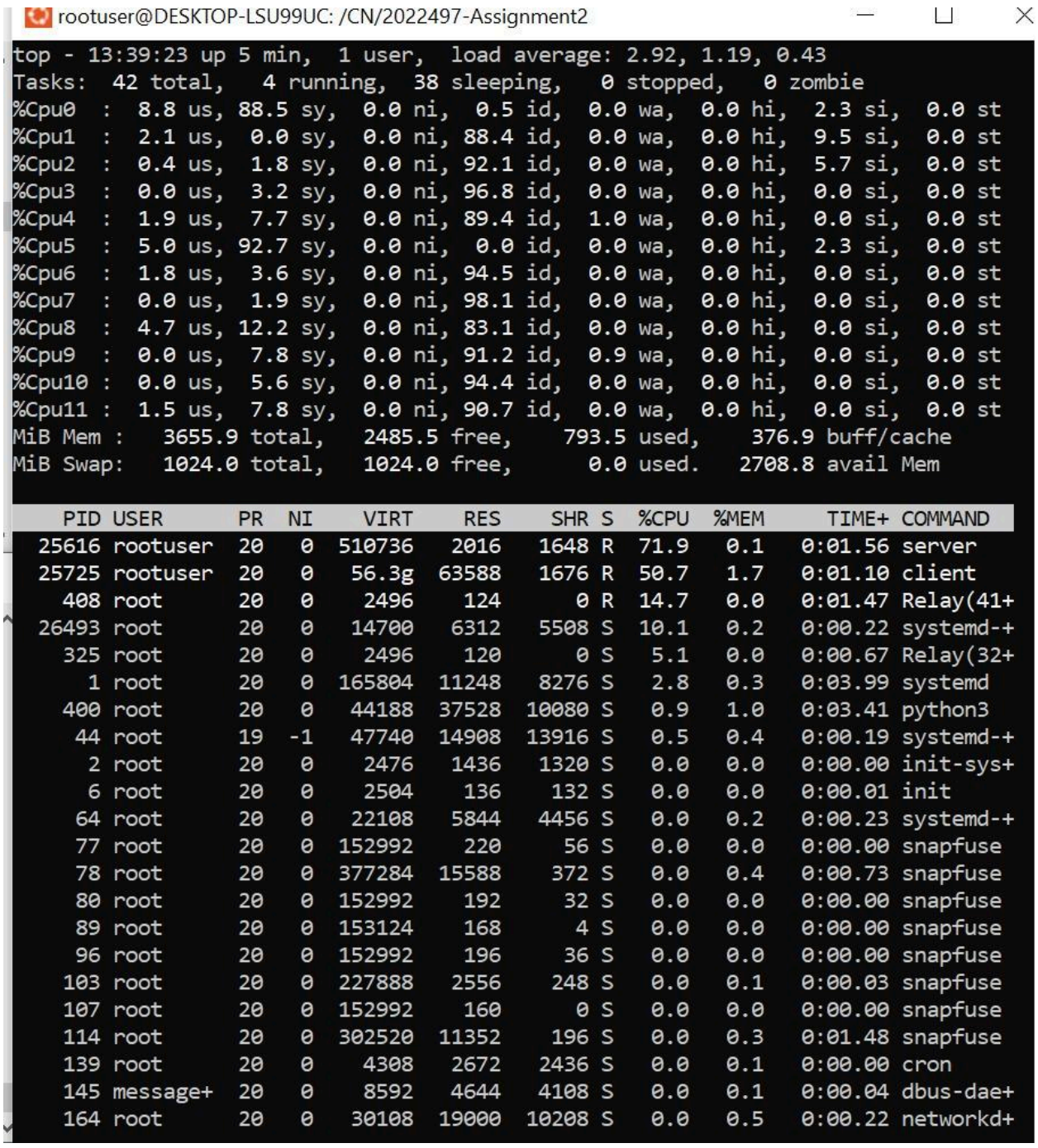


Client and server running:





CPU configurations during running client and server:



Performance Analysis-

Server is running with PID 25616

Client is running with PID 25725

Server running on CPU0 with cpu utilization of 71.9 %

Client running on CPU5 with cpu utilization of 50.7 %

This suggests that computation power is high as they utilize a large amount of CPU for their processes. Client is running for 10000 threads. Overhead to manage these thousands of threads.Also they are pinned to a specific CPU so scheduler can not change CPUs of the server and client.

Question2

Perf for part 2A:

```
root@DESKTOP-8METIQG:/mnt/d/Shubham/CN_Assignment2/CN_Assignment2/2A# perf stat -e cycles,instructions,branches,branch-m
isses,task-clock,context-switches,cpu-migrations,page-faults,stalled-cycles-frontend,stalled-cycles-backend  ./client 2
Client 1 message send to server
Client 1 Response form server: Top 2 CPU-consuming processes:
1. PID: 305, Name: (python3.10), User CPU Time: 254, Kernel CPU Time: 35, Total CPU Time: 289
2. PID: 132, Name: (snapfuse), User CPU Time: 71, Kernel CPU Time: 29, Total CPU Time: 100

Client 2 message send to server
Client 2 Response form server: Top 2 CPU-consuming processes:
1. PID: 305, Name: (python3.10), User CPU Time: 254, Kernel CPU Time: 35, Total CPU Time: 289
2. PID: 132, Name: (snapfuse), User CPU Time: 71, Kernel CPU Time: 29, Total CPU Time: 100


 Performance counter stats for './client 2':

          2121050      cycles                    #    0.844 GHz
           884473      instructions              #    0.42  insn per cycle
           185678      branches                  #   73.890 M/sec
            11068      branch-misses             #    5.96% of all branches
             2.51 msec task-clock                #    0.142 CPUs utilized
               11      context-switches          #    4.377 K/sec
                1      cpu-migrations            #  397.947 /sec
               61      page-faults               #   24.275 K/sec
```

Perf for part 2B:

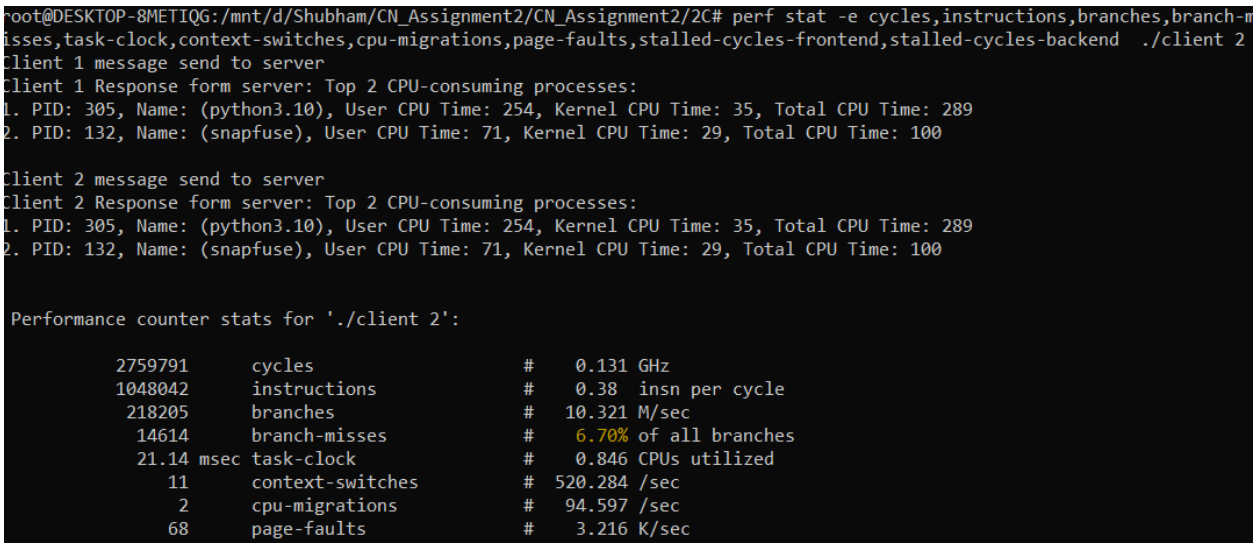
```
root@DESKTOP-8METIQG:/mnt/d/Shubham/CN_Assignment2/CN_Assignment2/2B# perf stat -e cycles,instructions,branches,branch-m
isses,task-clock,context-switches,cpu-migrations,page-faults,stalled-cycles-frontend,stalled-cycles-backend  ./client 2
Client 1 message send to server
Client 2 message send to server
Client 1 Response form server: Top 2 CPU-consuming processes:
1. PID: 305, Name: (python3.10), User CPU Time: 254, Kernel CPU Time: 35, Total CPU Time: 289
2. PID: 132, Name: (snapfuse), User CPU Time: 71, Kernel CPU Time: 29, Total CPU Time: 100

Client 2 Response form server: Top 2 CPU-consuming processes:
1. PID: 305, Name: (python3.10), User CPU Time: 254, Kernel CPU Time: 35, Total CPU Time: 289
2. PID: 132, Name: (snapfuse), User CPU Time: 71, Kernel CPU Time: 29, Total CPU Time: 100


 Performance counter stats for './client 2':

          3119600      cycles                    #    0.569 GHz
          1035075      instructions              #    0.33  insn per cycle
           217935      branches                  #   39.747 M/sec
            14389      branch-misses             #    6.60% of all branches
             5.48 msec task-clock                #    0.270 CPUs utilized
               13      context-switches          #    2.371 K/sec
                3      cpu-migrations            #  547.146 /sec
               66      page-faults               #   12.037 K/sec
```

Perf for part 2C:



```
root@DESKTOP-8METIQG:/mnt/d/Shubham/CN_Assignment2/CN_Assignment2/2C# perf stat -e cycles,instructions,branches,branch-m
isses,task-clock,context-switches,cpu-migrations,page-faults,stalled-cycles-frontend,stalled-cycles-backend  ./client 2
Client 1 message send to server
Client 1 Response form server: Top 2 CPU-consuming processes:
1. PID: 305, Name: (python3.10), User CPU Time: 254, Kernel CPU Time: 35, Total CPU Time: 289
2. PID: 132, Name: (snapfuse), User CPU Time: 71, Kernel CPU Time: 29, Total CPU Time: 100

Client 2 message send to server
Client 2 Response form server: Top 2 CPU-consuming processes:
1. PID: 305, Name: (python3.10), User CPU Time: 254, Kernel CPU Time: 35, Total CPU Time: 289
2. PID: 132, Name: (snapfuse), User CPU Time: 71, Kernel CPU Time: 29, Total CPU Time: 100


 Performance counter stats for './client 2':

          2759791      cycles                    #    0.131 GHz
          1048042      instructions              #    0.38  insn per cycle
           218205      branches                  #   10.321 M/sec
            14614      branch-misses             #    6.70% of all branches
            21.14 msec task-clock                #    0.846 CPUs utilized
               11      context-switches          #  520.284 /sec
                2      cpu-migrations            #   94.597 /sec
               68      page-faults               #    3.216 K/sec
```

Metric	Single-threaded	Concurrent	Using "select"
Cycles	2,121,050	3,119,600	2,759,791
Instructions	884,473	1,035,075	1,048,042
Instructions per Cycle	0.42	0.33	0.38
Branches	185,678	217,935	218,205
Branch Misses	11,068 (5.96%)	14,389 (6.60%)	14,614 (6.70%)
Task Clock (ms)	2.51	5.48	21.14
CPU Utilization	0.142 CPUs	0.270 CPUs	0.846 CPUs
Context Switches	1	13	305
CPU Migrations	1	3	2
Page Faults	61	66	68
GHz	0.844 GHz	0.569 GHz	0.131 GHz

Performance Analysis:

1. **Instructions per Cycle (IPC):**

   Single-threaded: Has the highest IPC at 0.42, indicating better CPU efficiency.

   Concurrent: Shows the lowest IPC at 0.33, likely due to the overhead of managing multiple threads.

   Using "select": Slightly better than the concurrent method at 0.38, showing improved efficiency in handling multiple connections.

2. **Branch Misses:**

   The **single-threaded** method has the lowest branch miss rate (5.96%), while both the **concurrent** (6.60%) and **select** (6.70%) approaches show increased mispredictions, likely due to handling more complex branching patterns.

3. **Task Clock and CPU Utilization:**

   Select-based method consumes significantly more CPU (0.846 CPUs utilized) and task clock time (21.14 ms) compared to the other methods. This indicates that while it is effective at managing multiple connections, it comes at the cost of higher resource usage.

   Single-threaded shows minimal CPU usage and task clock time, being more efficient in terms of processing single requests.

4. **Context Switches and CPU Migrations:**

   Using "select" has the highest number of context switches (305) compared to 13 in the **concurrent** and only 1 in the **single-threaded** version. This overhead is due to managing multiple file descriptors, which leads to frequent switching.

   The number of CPU migrations is low across all methods, with a slight increase in the **concurrent** method (3 migrations) compared to the **single-threaded** and **select** approaches.

5. **Page Faults:**

   The number of page faults is similar across all methods, with **single-threaded** having the lowest at 61 and the **select** method having the highest at 68. This indicates that memory access patterns remain relatively stable across methods.