CN Assignment 2 - Report

Q1.

In this question we have set up a TCP connection between the server and the client. It has been done using **‘tasknet’** to pin server to **CPU core 0** while pinning client to **CPU core 1**. The server-client connection being multithreaded in nature, the client requires an argument for a number of concurrent connections to be made.

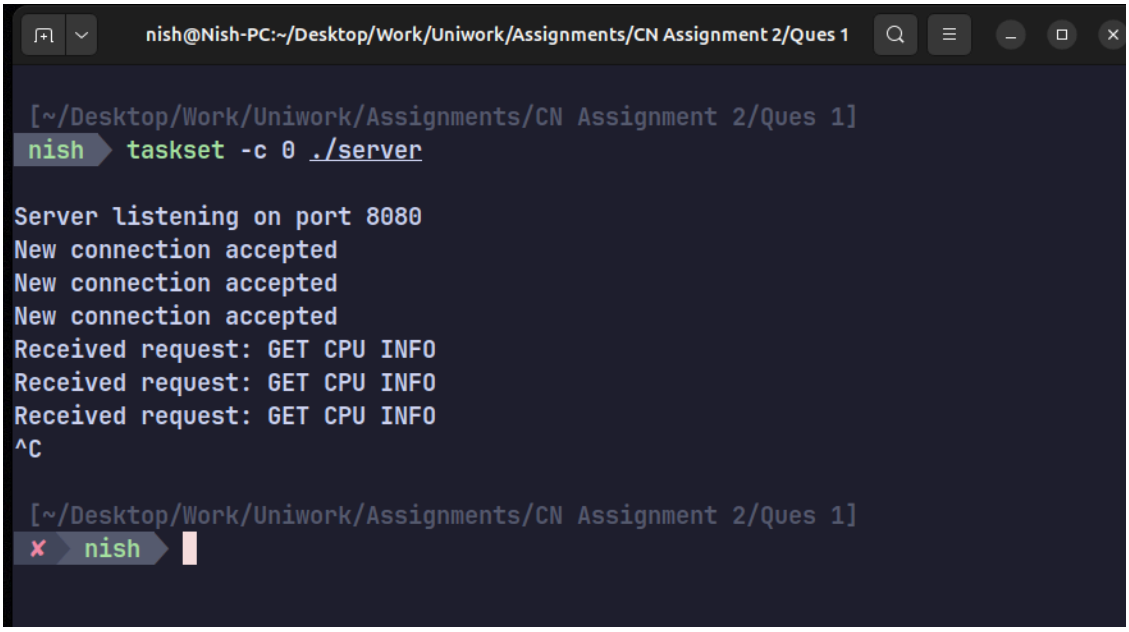


Fig. 1

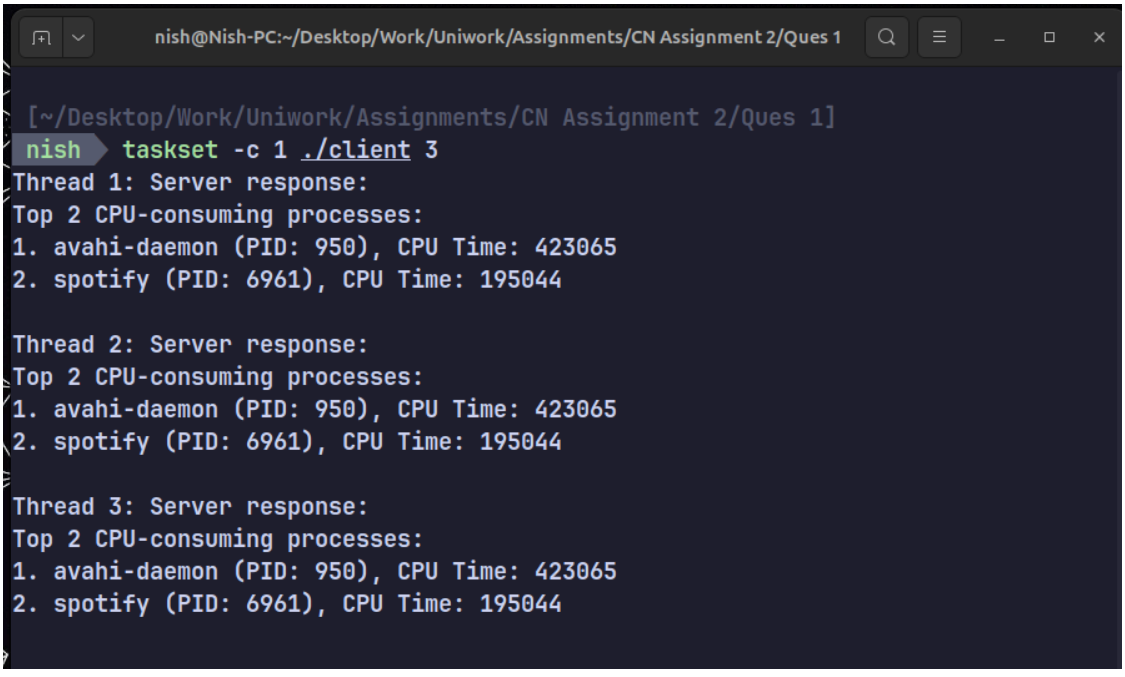


Fig. 2

The port being default at **8080** and the number of connections made by the multithreaded client being **3**, we can see the response from the server along with acknowledgement of receival of the request and departure of the response in Fig. 1.

On the client terminal we can see the response from the server received by the 3 threads, where we can see the top 2 processes in terms of CPU time along with their name and PIDs. They are **avahi-daemon (PID : 950)** and **spotify (PID : 6961)**. The clients close their socket after receiving the reply back from the server and the client is terminated. We can also see the Ids of thread shown along with the server response.

The data for all these processes is gathered using the saved information about the process in **/proc/[pid]/stat** in linux filesystem, where it was filtered using the source code and sent across the connection.

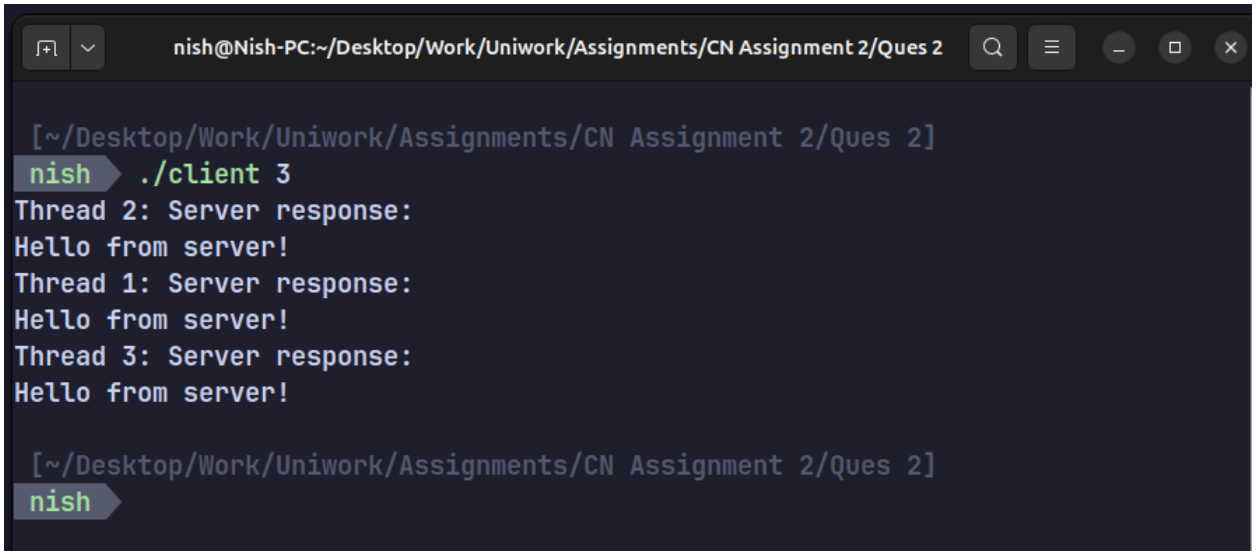
The server is terminated using the SIGINT (ctrl + c) to shutdown the server for any more incoming requests.

Q2.

The following are the performance statistics measured for
(a) Single-threaded TCP client-server
(b) Concurrent TCP client-server
(c) TCP client-server using “select”
using the **‘perf’** tool, meanwhile pinning the server to **CPU core 0** and client to **CPU core 1**, using **‘tasknet’** to reduce the migrations and context switches for the CPU. Below is a comprehensive comparison of the listed connections.

(a) Single-threaded TCP client-server

Client



Server

```
[../Work/Uniwork/Assignments/CN Assignment 2/Ques 2/Single Thread]
nish  sudo taskset -c 0 perf stat ./singleThreadServer
Waiting for connections...
Client request: Incoming Client Request
Response sent
Client request: Incoming Client Request
Response sent
Client request: Incoming Client Request
Response sent
^C./singleThreadServer: Interrupt

 Performance counter stats for './singleThreadServer':

              0.54 msec task-clock                       #    0.000 CPUs utilized
                 2      context-switches                 #    3.685 K/sec
                 0      cpu-migrations                   #    0.000 /sec
                54      page-faults                      #   99.503 K/sec
    <not counted>      cpu_atom/cycles/                                              (0.00%)
         1,311,397      cpu_core/cycles/                 #    2.416 GHz
    <not counted>      cpu_atom/instructions/                                        (0.00%)
         1,199,173      cpu_core/instructions/
    <not counted>      cpu_atom/branches/                                            (0.00%)
           215,750      cpu_core/branches/               #  397.552 M/sec
    <not counted>      cpu_atom/branch-misses/                                       (0.00%)
             8,301      cpu_core/branch-misses/
             TopdownL1 (cpu_core)                 #     20.0 %  tma_backend_bound
                                                  #     10.8 %  tma_bad_speculation
                                                  #     50.0 %  tma_frontend_bound
                                                  #     19.2 %  tma_retiring

       8.411382083 seconds time elapsed

       0.000000000 seconds user
       0.000794000 seconds sys
```

(b) Concurrent TCP client-server

Client

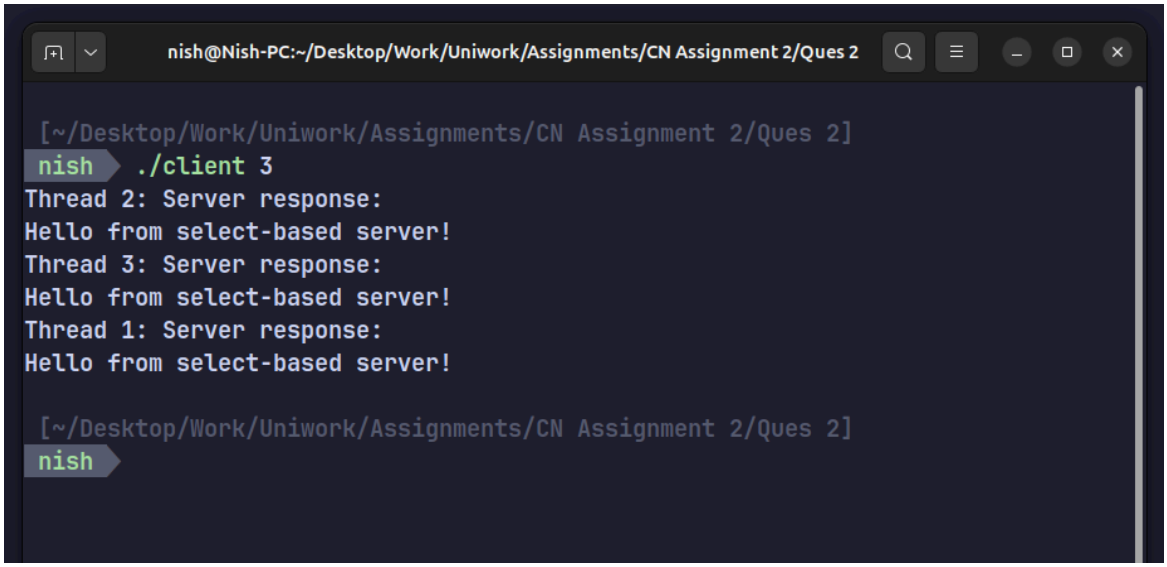
```
nish@Nish-PC:~/Desktop/Work/Uniwork/Assignments/CN Assignment 2/Ques 2

 [~/Desktop/Work/Uniwork/Assignments/CN Assignment 2/Ques 2]
nish  ./client 3
Thread 1: Server response:
Hello from multi-threaded server!
Thread 2: Server response:
Hello from multi-threaded server!
Thread 3: Server response:
Hello from multi-threaded server!

 [~/Desktop/Work/Uniwork/Assignments/CN Assignment 2/Ques 2]
nish
```

Server

```
[..p/Work/Uniwork/Assignments/CN Assignment 2/Ques 2/Multi Thread]
nish  sudo taskset -c 0 perf stat ./multiThreadServer
Waiting for connections...
Client request: Incoming Client Request
Response sent
Client request: Incoming Client Request
Response sent
Client request: Incoming Client Request
Response sent
^C./multiThreadServer: Interrupt

 Performance counter stats for './multiThreadServer':

             1.16 msec task-clock                       #    0.000 CPUs utilized
                2      context-switches                 #    1.727 K/sec
                0      cpu-migrations                   #    0.000 /sec
               66      page-faults                      #   56.985 K/sec
    <not counted>      cpu_atom/cycles/                                              (0.00%)
        1,911,324      cpu_core/cycles/                 #    1.650 GHz
    <not counted>      cpu_atom/instructions/                                        (0.00%)
        1,803,031      cpu_core/instructions/
    <not counted>      cpu_atom/branches/                                            (0.00%)
          328,856      cpu_core/branches/               #  283.940 M/sec
    <not counted>      cpu_atom/branch-misses/                                       (0.00%)
           12,244      cpu_core/branch-misses/
            TopdownL1 (cpu_core)                 #     21.5 %  tma_backend_bound
                                                 #     10.3 %  tma_bad_speculation
                                                 #     50.1 %  tma_frontend_bound
                                                 #     18.1 %  tma_retiring

      8.693918827 seconds time elapsed

      0.000000000 seconds user
      0.001301000 seconds sys
```

(c) TCP client-server using “select”

Client



Server

```
[../Work/Uniwork/Assignments/CN Assignment 2/Ques 2/Select Thread]
nish  sudo taskset -c 0 perf stat ./selectServer
Waiting for connections...
Client request: Incoming Client Request
Response sent
Client request: Incoming Client Request
Response sent
Client request: Incoming Client Request
Response sent
^C./selectServer: Interrupt

 Performance counter stats for './selectServer':

              0.77 msec task-clock                       #    0.000 CPUs utilized
                 2      context-switches                 #    2.581 K/sec
                 0      cpu-migrations                   #    0.000 /sec
                54      page-faults                      #   69.698 K/sec
     <not counted>      cpu_atom/cycles/                                              (0.00%)
         1,456,055      cpu_core/cycles/                 #    1.879 GHz
     <not counted>      cpu_atom/instructions/                                        (0.00%)
         1,200,848      cpu_core/instructions/
     <not counted>      cpu_atom/branches/                                            (0.00%)
           216,446      cpu_core/branches/               #  279.368 M/sec
     <not counted>      cpu_atom/branch-misses/                                       (0.00%)
             8,729      cpu_core/branch-misses/
             TopdownL1 (cpu_core)                 #     23.0 %  tma_backend_bound
                                                  #      9.5 %  tma_bad_speculation
                                                  #     51.8 %  tma_frontend_bound
                                                  #     15.6 %  tma_retiring

       2.930543819 seconds time elapsed

       0.000000000 seconds user
       0.001199000 seconds sys
```

From the above results we can clearly see that,

Single-Threaded Server:

- **Task Clock**: 0.54 msec | **Elapsed Time**: 8.41 sec | **CPU Cycles**: 1,311,397
- The connection has a very minimal CPU time utilization, having the lowest task clock but also the longest total elapsed time which is due to handling multiple concurrent clients having a single threaded nature which results in above slow metrics. Despite a relatively efficient instruction handling (19.2% of instructions retiring), it still lacks in performance due to single-threaded execution.

Select-Based Server:

- **Task Clock**: 0.77 msec | **Elapsed Time**: 2.93 sec | **CPU Cycles**: 1,456,055
- The optimal balance between performance and resource utilization is provided by this connection. It effectively manages several clients with a moderate task clock and CPU cycle count by using the **select** system call to control connections. Its capacity to handle concurrent client requests without the expense of threading is demonstrated by the shorter elapsed time. The TMA measurements show that it is best suited for scalable applications with modest loads because it can handle frontend-bound activities (51.8%) efficiently and with low backend-bound delay.

Multi-Threaded Server:

- **Task Clock**: 1.16 msec | **Elapsed Time**: 8.69 sec | **CPU Cycles**: 1,911,324
- The multi-threaded server uses many threads to process client requests in parallel, although thread management adds to the server's overhead. The increased task clock and CPU cycle count demonstrate this. Despite providing genuine parallelism, the complexity of managing numerous threads results in a very large elapsed time. The TMA metrics indicate a marginally greater frontend-bound workload (50.1%) and backend-bound operation (21.5%), indicating that although it is required in some high-load situations, it is less effective than the choose model at managing numerous client requests at once.

These results are further consolidated in the table given below.

Metric	Single-Threaded Server	Select-Based Server	Multi-Threaded Server
Task-Clock	0.54 msec	0.77 msec	1.16 msec
Context Switches	2	2	2
CPU Migrations	0	0	0
Page Faults	54	54	66
CPU Cycles	1,311,397	1,456,055	1,911,324
Instructions	1,199,173	1,200,848	1,803,031
Branches	215,750	216,446	328,856
Branch Misses	8,301	8,729	12,244
Elapsed Time	8.41 seconds	2.93 seconds	8.69 seconds
User Time	0.000000000 seconds	0.000000000 seconds	0.000000000 seconds
System Time	0.000794000 seconds	0.001199000 seconds	0.001301000 seconds
TMA Backend Bound (%)	20.0%	23.0%	21.5%
TMA Bad Speculation (%)	10.8%	9.5%	10.3%
TMA Frontend Bound (%)	50.0%	51.8%	50.1%
TMA Retiring (%)	19.2%	15.6%	18.1%

Contributors

1. Nishchay Yadav - 2022332
2. Nishant Yadav - 2022329

Github Repo : [CN Assignment 2 Github Repo]