CSE 232: Programming Assignment 2
Socket programming with performance analysis

Due date: Sep 30, 2024

Total: 20 points

Read the following instructions carefully

- For all the observations and explanations, create a single report.
- Attach screenshots in the report.
- Naming Convention: <Roll_No>-Assignment2.zip
- Create a public git repository for the course, and add your report and code (if applicable).
- This assignment is pair-wise. The pair is strictly as per your registration number in the shared sheet. For example, Sr no. 1 & 2, 3 & 4, 5 & 6, ….. ,171 & 172 forms a pair.

Q.1. Write a client-server socket program in C. The client process and the server process should run on separate VMs (or containers) and communicate with each other. Use "taskset" to pin the process to specific CPUs. This helps measure the performance. Your program should have the following features. [1+(1+2)+2+(1+2)+1+1]

1. The server sets up a TCP socket and listens for client connections.
2. The server should be able to handle multiple concurrent clients (multithreaded, concurrent server). The server accepts the client connection; hence, a new socket is created with 4 tuples (server IP, server listening port, client IP, client port). The server creates a new thread that continues to process the client connection (Hint: Use pthread library for multithreading). The original server socket continues to listen on the same listening port for newer incoming client connections.
3. The client creates a socket and initiates the TCP connection. Your client process should support initiating "n" concurrent client connection requests, where "n" is passed as a program argument.
4. After the client connection is established, the client sends a request to the server to get information about the server's top TWO CPU-consuming processes. The server finds out the top CPU-consuming process (user+kernel CPU time) and gathers information such as process name, pid (process id), and CPU usage of the process in user & kernel mode (you can report this time in clock ticks).
    - **Possible solution approach:** The server should make use of the open() system call to read the "proc" filesystem to get this information. You need to read ***/proc/[pid]/stat*** for all processes to parse the process name, pid, and CPU time (user+kernel). To understand the format of ***/proc/[pid]/stat*** file, refer the source code of show_stat function is available here for reference. More about "proc" file system is here.
5. Server sends the information collected in **step (4)** to the client.
6. The client prints this information & closes the connection.

Q.2. The socket programming source code that leverages "select" system call here. Modify the server code as per Q.1. Use the perf tool to analyze the performance of the following: **[3+3+3]**

(a) Single-threaded TCP client-server
(b) Concurrent TCP client-server
(c) TCP client-server using "select"

You can be creative for this analysis. You may take readings of various performance counters (CPU clocks, cache misses, context switches, etc.) across number of concurrent connections, etc. This is an OPEN question; you can be as comprehensive as you can.

Q.3. Bonus question —> For the ones who want to learn more. Does NOT carry marks

Repeat Q.2. for 'poll" and "epoll" as well for the analysis. I will personally take demo for this part.

Sample format of ***/proc/[pid]/stat***
(Details at https://man7.org/linux/man-pages/man5/proc.5.html & *https://github.com/torvalds/linux/blob/master/fs/proc/stat.c* ***)***
User space and kernel space CPU time (measured time in clock ticks) is marked in **RED**.

```
$ cat /proc/76893/stat
76893 (top) S 76734 76893 76734 34825 76893 4194304 400 0 0 0 573 806 0 0 20 0 1 0
1453568921 43216896 1066 18446744073709551615 4194304 4292532 140726554046848 0 0 0 0
0 2147155711 1 0 0 17 15 0 0 0 0 0 6393312 6398352 14311424 140726554052535
140726554052539 140726554052539 140726554054635
```