CSE 232 : Programming Assignment 2

Socket programming with performance analysis

Yash Singh & *Bhavy Chawla*

Q1.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. The server sets up a TCP socket and listens for client connections.

```c
void setupServerSocket() {
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    int opt = 1;
    setsockopt(serverSocket, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

    struct sockaddr_in address;
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(serverSocket, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        close(serverSocket);
        exit(EXIT_FAILURE);
    }

    if (listen(serverSocket, MAX_CLIENTS) < 0) {
        perror("Listen failed");
        close(serverSocket);
        exit(EXIT_FAILURE);
    }
}
```

2.The server accepts client connections to create new sockets while spawning new threads to process each connection and continuing to listen for additional clients on the same port.

```c
void acceptClients() {
    int addrlen = sizeof(struct sockaddr_in);
    struct sockaddr_in address;

    while (1) {
        int newSocket = accept(serverSocket, (struct sockaddr *)&address, (socklen_t *)&addrlen);
        if (newSocket < 0) {
            perror("Accept failed");
            exit(EXIT_FAILURE);
        }

        printf("Client connected from %s:%d\n", inet_ntoa(address.sin_addr), ntohs(address.sin_port));

        pthread_t clientThread;
        int *newSock = malloc(1);
        *newSock = newSocket;

        if (pthread_create(&clientThread, NULL, handleClient, (void *)newSock) < 0) {
            perror("Could not create thread");
            free(newSock);
            exit(EXIT_FAILURE);
        }

        pthread_detach(clientThread);
    }
}
```

3.The client creates a socket to initiate a TCP connection, supporting "n" concurrent client connection requests specified as a program argument.

```c
void createClientThreads(int numClients) {
    pthread_t clientThreads[numClients];
    int threadIds[numClients];

    for (int i = 0; i < numClients; i++) {
        threadIds[i] = i;
        if (pthread_create(&clientThreads[i], NULL, clientTask, (void *)&threadIds[i]) != 0) {
            perror("Could not create client thread");
            fclose(logFile);
            exit(1);
        }
    }

    for (int i = 0; i < numClients; i++) {
        pthread_join(clientThreads[i], NULL);
    }
}
```

4.After establishing the client connection, the client requests the server for information on the top two CPU-consuming processes, and the server retrieves their process name, PID, and CPU usage (user and kernel time in clock ticks) by using the `open()` system call to read the `/proc/[pid]/stat` files for all processes.

```c
void *clientTask(void *arg) {
    int clientSocket = 0;
    struct sockaddr_in serverAddress;
    char receiveBuffer[BUFFER_SIZE] = {0};

    if ((clientSocket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        logMessage("Socket creation error\n");
        return NULL;
    }

    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serverAddress.sin_addr) <= 0) {
        logMessage("Invalid address\n");
        close(clientSocket);
        return NULL;
    }
    if (connect(clientSocket, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) < 0) {
        logMessage("Connection failed\n");
        close(clientSocket);
        return NULL;
    }
    int bytesRead = read(clientSocket, receiveBuffer, BUFFER_SIZE);
    if (bytesRead > 0) {
        pthread_mutex_lock(&logMutex);
        totalResponseCount++;
        char logEntry[BUFFER_SIZE + 100];
        snprintf(logEntry, sizeof(logEntry), "Thread ID: %ld, Response Number: %d\nServer Response:\n%s\n",
                 pthread_self(), totalResponseCount, receiveBuffer);
        logMessage(logEntry);
        pthread_mutex_unlock(&logMutex);
    } else {
        logMessage("Error reading from server\n");
    }
    close(clientSocket);
    return NULL;
}
```

Server side

```c
struct ProcessInfo {
    char name[256];
    int pid;
    unsigned long cpuUser;
    unsigned long cpuKernel;
};

// Function to parse the content of /proc/[pid]/stat using open() and read()
void extractProcessInfo(int pid, struct ProcessInfo *process) {
    char path[256], buffer[BUFFER_SIZE];
    int fd, bytesRead;

    sprintf(path, "/proc/%d/stat", pid);
    fd = open(path, O_RDONLY);
    if (fd == -1) return;

    bytesRead = read(fd, buffer, sizeof(buffer) - 1);
    if (bytesRead <= 0) {
        close(fd);
        return;
    }
    buffer[bytesRead] = '\0';
    close(fd);

    sscanf(buffer, "%d %s %*c %*d %*d %*d %*d %*d %*u %*u %*u %*u %*u %lu %lu",
           &process->pid, process->name, &process->cpuUser, &process->cpuKernel);
}
```

```c
void getTopTwoProcesses(struct ProcessInfo *topProcesses) {
    unsigned long maxCpu1 = 0, maxCpu2 = 0;
    struct ProcessInfo process1 = {0}, process2 = {0}, currentProcess = {0};

    DIR *dir;
    struct dirent *entry;

    dir = opendir("/proc");
    if (dir == NULL) {
        perror("Unable to open /proc directory");
        return;
    }

    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_DIR && atoi(entry->d_name) > 0) {
            extractProcessInfo(atoi(entry->d_name), &currentProcess);
            unsigned long totalCpu = currentProcess.cpuUser + currentProcess.cpuKernel;

            if (totalCpu > maxCpu1) {
                process2 = process1;
                maxCpu2 = maxCpu1;
                process1 = currentProcess;
                maxCpu1 = totalCpu;
            } else if (totalCpu > maxCpu2) {
                process2 = currentProcess;
                maxCpu2 = totalCpu;
            }
        }
    }

    closedir(dir);
    topProcesses[0] = process1;
    topProcesses[1] = process2;
}
```

5.Server sends the information collected in step (4) to the client.

```c
void sendProcessInfo(int clientSocket) {
    struct ProcessInfo topProcesses[2];
    getTopTwoProcesses(topProcesses);

    char buffer[BUFFER_SIZE];
    sprintf(buffer, "Process 1: %s (PID: %d, CPU User: %lu, CPU Kernel: %lu)\n"
                    "Process 2: %s (PID: %d, CPU User: %lu, CPU Kernel: %lu)\n",
            topProcesses[0].name, topProcesses[0].pid, topProcesses[0].cpuUser, topProcesses[0].cpuKernel,
            topProcesses[1].name, topProcesses[1].pid, topProcesses[1].cpuUser, topProcesses[1].cpuKernel);

    send(clientSocket, buffer, strlen(buffer), 0);
    __sync_fetch_and_add(&totalResponsesSent, 1);
}
```

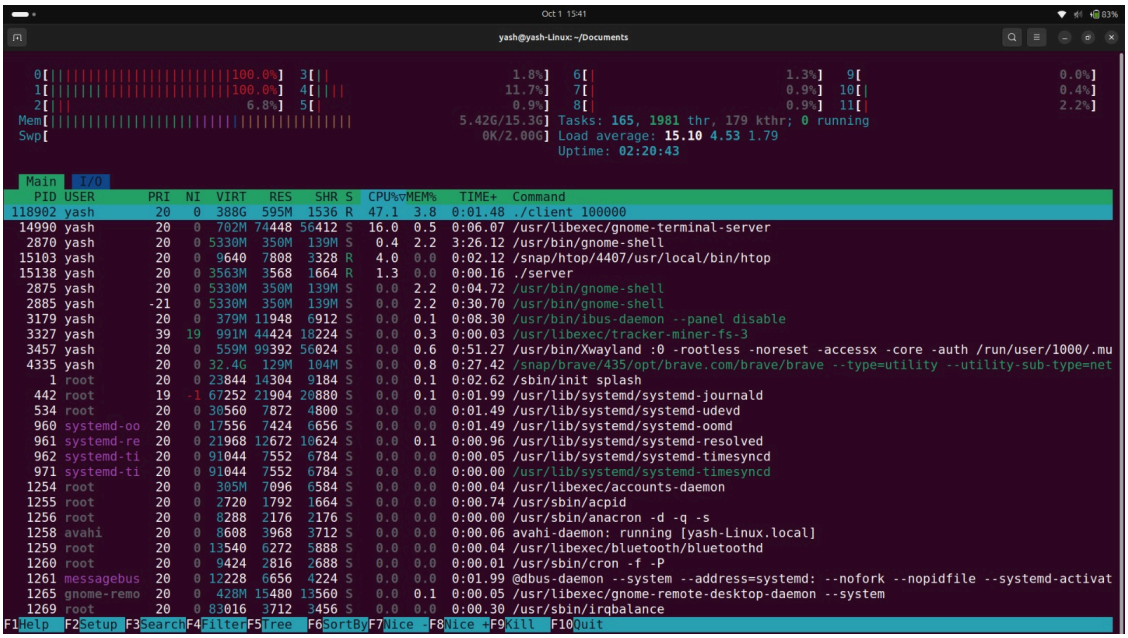
6.The client prints this information & closes the connection.

```c
int bytesRead = read(clientSocket, receiveBuffer, BUFFER_SIZE);
if (bytesRead > 0) {
    pthread_mutex_lock(&logMutex);
    totalResponseCount++;
    char logEntry[BUFFER_SIZE + 100];
    snprintf(logEntry, sizeof(logEntry), "Thread ID: %ld, Response Number: %d\nServer Response:\n%s\n",
             pthread_self(), totalResponseCount, receiveBuffer);
    logMessage(logEntry);
    pthread_mutex_unlock(&logMutex);
} else {
    logMessage("Error reading from server\n");
}

close(clientSocket);
return NULL;
```

```
yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/B_Concurrent/Client$ perf stat taskset -c 0 ./client 100
Thread ID: 126446584465088, Response Number: 1
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9046, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126446878066368, Response Number: 2
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9046, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126446888552128, Response Number: 3
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9046, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126447433811648, Response Number: 4
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9045, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126446794180288, Response Number: 5
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9046, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126447402354368, Response Number: 6
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9045, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126447266039488, Response Number: 7
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9046, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)

Thread ID: 126447465268928, Response Number: 8
Server Response:
```

```
Thread ID: 126446563493568, Response Number: 100
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9047, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)


 Performance counter stats for 'taskset -c 0 ./client 100':

            16.04 msec task-clock                       #    0.004 CPUs utilized
               152      context-switches                 #    9.476 K/sec
                 1      cpu-migrations                   #   62.345 /sec
               442      page-faults                      #   27.557 K/sec
        88,97,987      cpu_atom/cycles/                 #    0.555 GHz                      (13.03%)
     5,51,49,045      cpu_core/cycles/                 #    3.438 GHz                      (97.75%)
        74,90,144      cpu_atom/instructions/           #    0.84  insn per cycle           (13.03%)
     3,30,58,689      cpu_core/instructions/           #    3.72  insn per cycle           (97.75%)
        13,58,618      cpu_atom/branches/               #   84.703 M/sec                    (13.03%)
        60,45,432      cpu_core/branches/               #  376.904 M/sec                    (97.75%)
           58,539      cpu_atom/branch-misses/          #    4.31% of all branches          (13.03%)
         1,00,408      cpu_core/branch-misses/          #    7.39% of all branches          (97.75%)
            TopdownL1 (cpu_core)                 #     22.6 %  tma_backend_bound
                                                 #      4.3 %  tma_bad_speculation
                                                 #     54.0 %  tma_frontend_bound
                                                 #     19.2 %  tma_retiring             (97.75%)
            TopdownL1 (cpu_atom)                 #     45.0 %  tma_bad_speculation
                                                 #     21.4 %  tma_retiring             (13.03%)
                                                 #      0.0 %  tma_backend_bound
                                                 #      0.0 %  tma_backend_bound_aux
                                                 #     33.5 %  tma_frontend_bound       (13.03%)

       3.739714213 seconds time elapsed

       0.000716000 seconds user
       0.008240000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/B_Concurrent/Client$
```

```
^C
Caught signal 2. Closing server socket and exiting...

Analytics Report:
Total responses sent: 100
Server socket closed.

 Performance counter stats for 'taskset -c 1 ./server':

          3,718.90 msec task-clock                       #    0.057 CPUs utilized
             1,557      context-switches                 #  418.672 /sec
                 1      cpu-migrations                   #    0.269 /sec
               631      page-faults                      #  169.674 /sec
       3,94,75,579      cpu_atom/cycles/                 #    0.011 GHz                      (1.02%)
    15,64,62,22,695      cpu_core/cycles/                 #    4.207 GHz                      (99.99%)
       3,13,29,337      cpu_atom/instructions/           #    0.79  insn per cycle           (3.14%)
    33,73,65,34,341      cpu_core/instructions/           #  854.62  insn per cycle           (99.99%)
          56,57,031      cpu_atom/branches/               #    1.521 M/sec                    (3.14%)
     6,00,67,24,735      cpu_core/branches/               #    1.615 G/sec                    (99.99%)
          2,49,700      cpu_atom/branch-misses/          #    4.41% of all branches          (3.14%)
       2,57,01,827      cpu_core/branch-misses/          #  454.33% of all branches          (99.99%)
            TopdownL1 (cpu_core)                 #     23.1 %  tma_backend_bound
                                                 #      1.9 %  tma_bad_speculation
                                                 #     35.5 %  tma_frontend_bound
                                                 #     39.5 %  tma_retiring             (99.99%)
            TopdownL1 (cpu_atom)                 #     17.2 %  tma_bad_speculation
                                                 #     21.2 %  tma_retiring             (3.14%)
                                                 #     28.6 %  tma_backend_bound
                                                 #     28.6 %  tma_backend_bound_aux
                                                 #     32.9 %  tma_frontend_bound       (3.14%)

      64.924246524 seconds time elapsed

       0.858277000 seconds user
       2.801936000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/B_Concurrent/Server$
```



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:

(a) Single-threaded TCP client-server

Server

home > yash > Documents > CN_ASSIGNMENT_2 > Q2 > A_SingleThread > Server > C server.c > main()

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    const char *hello = "Hello from server";
    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }
    // Defining address for the server
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(8002);
    // Binding the socket to the port
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }
    // Start listening for incoming connections
    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        exit(EXIT_FAILURE);
    }
    printf("Server is listening on port 8002...\n");
    // Accepting an incoming connection
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
        perror("Accept failed");
        exit(EXIT_FAILURE);
    }
    // Reading the incoming message from the client
    read(new_socket, buffer, 1024);
    printf("Message received: %s\n", buffer);
    // Sending a message back to the client
    send(new_socket, hello, strlen(hello), 0);
    printf("Hello message sent\n");
    // Closing the socket
    close(new_socket);
    close(server_fd);
    return 0;
}
```

0 ⚠ 0 0

Client

home > yash > Documents > CN_ASSIGNMENT_2 > Q2 > A_SingleThread > Client > C client.c > main()

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    const char *hello = "Hello from client";
    char buffer[1024] = {0};

    // Creating socket
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation error");
        return -1;
    }

    // Setting up the server address structure
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(8002);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        perror("Invalid address/Address not supported");
        return -1;
    }

    // Connecting to the server
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connection failed");
        return -1;
    }

    // Sending a message to the server
    send(sock, hello, strlen(hello), 0);
    printf("Hello message sent\n");

    // Reading the server's response
    read(sock, buffer, 1024);
    printf("Message received: %s\n", buffer);

    // Closing the socket
    close(sock);

    return 0;
}
```

```
yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/A_SingleThread/Server$ perf stat taskset -c 1 ./server
Server is listening on port 8002...
Message received: Hello from client
Hello message sent

 Performance counter stats for 'taskset -c 1 ./server':

              0.81 msec task-clock                       #    0.000 CPUs utilized
                 2      context-switches                 #    2.471 K/sec
                 1      cpu-migrations                   #    1.235 K/sec
               127      page-faults                      #  156.882 K/sec
        25,25,789      cpu_atom/cycles/                 #    3.120 GHz                      (20.04%)
        32,94,052      cpu_core/cycles/                 #    4.069 GHz                      (54.49%)
        21,45,955      cpu_atom/instructions/           #    0.85  insn per cycle           (45.51%)
        28,23,088      cpu_core/instructions/           #    1.12  insn per cycle           (54.49%)
         3,88,931      cpu_atom/branches/               #  480.443 M/sec                    (45.51%)
         5,25,647      cpu_core/branches/               #  649.328 M/sec                    (54.49%)
           17,178      cpu_atom/branch-misses/          #    4.42% of all branches          (45.51%)
           17,267      cpu_core/branch-misses/          #    4.44% of all branches          (54.49%)
            TopdownL1 (cpu_core)                 #     25.9 %  tma_backend_bound
                                                 #     10.0 %  tma_bad_speculation
                                                 #     45.3 %  tma_frontend_bound
                                                 #     18.8 %  tma_retiring             (54.49%)
            TopdownL1 (cpu_atom)                 #     16.0 %  tma_bad_speculation
                                                 #     21.3 %  tma_retiring             (45.51%)
                                                 #     27.9 %  tma_backend_bound
                                                 #     27.9 %  tma_backend_bound_aux
                                                 #     34.8 %  tma_frontend_bound       (45.51%)

       2.632538255 seconds time elapsed

       0.000000000 seconds user
       0.001351000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/A_SingleThread/Server$
```

```
yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/A_SingleThread/Client$ perf stat taskset -c 0 ./client
Hello message sent
Message received: Hello from server

 Performance counter stats for 'taskset -c 0 ./client':

              0.95 msec task-clock                       #    0.535 CPUs utilized
                 2      context-switches                 #    2.106 K/sec
                 1      cpu-migrations                   #    1.053 K/sec
               127      page-faults                      #  133.746 K/sec
        27,05,709      cpu_atom/cycles/                 #    2.849 GHz                      (8.96%)
        36,14,850      cpu_core/cycles/                 #    3.807 GHz                      (54.39%)
        21,29,852      cpu_atom/instructions/           #    0.79  insn per cycle           (45.61%)
        29,10,804      cpu_core/instructions/           #    1.08  insn per cycle           (54.39%)
         3,87,101      cpu_atom/branches/               #  407.664 M/sec                    (45.61%)
         5,40,427      cpu_core/branches/               #  569.134 M/sec                    (54.39%)
           17,102      cpu_atom/branch-misses/          #    4.42% of all branches          (45.61%)
           17,864      cpu_core/branch-misses/          #    4.61% of all branches          (54.39%)
            TopdownL1 (cpu_core)                 #     29.0 %  tma_backend_bound
                                                 #      9.2 %  tma_bad_speculation
                                                 #     44.5 %  tma_frontend_bound
                                                 #     17.3 %  tma_retiring             (54.39%)
            TopdownL1 (cpu_atom)                 #     13.6 %  tma_bad_speculation
                                                 #     19.2 %  tma_retiring             (45.61%)
                                                 #     36.1 %  tma_backend_bound
                                                 #     36.1 %  tma_backend_bound_aux
                                                 #     31.1 %  tma_frontend_bound       (45.61%)

       0.001775354 seconds time elapsed

       0.000000000 seconds user
       0.001815000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/A_SingleThread/Client$
```

(b) Concurrent TCP client-server

```
Thread ID: 126446563493568, Response Number: 100
Server Response:
Process 1: (irq/148-rtw88_pci) (PID: 817, CPU User: 9047, CPU Kernel: 0)
Process 2: (systemd) (PID: 2583, CPU User: 12, CPU Kernel: 8899)


 Performance counter stats for 'taskset -c 0 ./client 100':

            16.04 msec task-clock                       #    0.004 CPUs utilized
              152      context-switches                 #    9.476 K/sec
                1      cpu-migrations                   #   62.345 /sec
              442      page-faults                      #   27.557 K/sec
       88,97,987      cpu_atom/cycles/                 #    0.555 GHz                      (13.03%)
     5,51,49,045      cpu_core/cycles/                 #    3.438 GHz                      (97.75%)
       74,90,144      cpu_atom/instructions/           #    0.84  insn per cycle           (13.03%)
     3,30,58,689      cpu_core/instructions/           #    3.72  insn per cycle           (97.75%)
       13,58,618      cpu_atom/branches/               #   84.703 M/sec                    (13.03%)
       60,45,432      cpu_core/branches/               #  376.904 M/sec                    (97.75%)
          58,539      cpu_atom/branch-misses/          #    4.31% of all branches          (13.03%)
        1,00,408      cpu_core/branch-misses/          #    7.39% of all branches          (97.75%)
           TopdownL1 (cpu_core)                 #     22.6 %  tma_backend_bound
                                                #      4.3 %  tma_bad_speculation
                                                #     54.0 %  tma_frontend_bound
                                                #     19.2 %  tma_retiring             (97.75%)
           TopdownL1 (cpu_atom)                 #     45.0 %  tma_bad_speculation
                                                #     21.4 %  tma_retiring             (13.03%)
                                                #      0.0 %  tma_backend_bound
                                                #      0.0 %  tma_backend_bound_aux
                                                #     33.5 %  tma_frontend_bound       (13.03%)

      3.739714213 seconds time elapsed

      0.000716000 seconds user
      0.008240000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/B_Concurrent/Client$
```

```
^C
Caught signal 2. Closing server socket and exiting...

Analytics Report:
Total responses sent: 100
Server socket closed.

 Performance counter stats for 'taskset -c 1 ./server':

         3,718.90 msec task-clock                       #    0.057 CPUs utilized
            1,557      context-switches                 #  418.672 /sec
                1      cpu-migrations                   #    0.269 /sec
              631      page-faults                      #  169.674 /sec
      3,94,75,579      cpu_atom/cycles/                 #    0.011 GHz                      (1.02%)
   15,64,62,22,695      cpu_core/cycles/                 #    4.207 GHz                      (99.99%)
      3,13,29,337      cpu_atom/instructions/           #    0.79  insn per cycle           (3.14%)
   33,73,65,34,341      cpu_core/instructions/           #  854.62  insn per cycle           (99.99%)
         56,57,031      cpu_atom/branches/               #    1.521 M/sec                    (3.14%)
    6,00,67,24,735      cpu_core/branches/               #    1.615 G/sec                    (99.99%)
          2,49,700      cpu_atom/branch-misses/          #    4.41% of all branches          (3.14%)
      2,57,01,827      cpu_core/branch-misses/          #  454.33% of all branches          (99.99%)
           TopdownL1 (cpu_core)                 #     23.1 %  tma_backend_bound
                                                #      1.9 %  tma_bad_speculation
                                                #     35.5 %  tma_frontend_bound
                                                #     39.5 %  tma_retiring             (99.99%)
           TopdownL1 (cpu_atom)                 #     17.2 %  tma_bad_speculation
                                                #     21.2 %  tma_retiring             (3.14%)
                                                #     28.6 %  tma_backend_bound
                                                #     28.6 %  tma_backend_bound_aux
                                                #     32.9 %  tma_frontend_bound       (3.14%)

     64.924246524 seconds time elapsed

      0.858277000 seconds user
      2.801936000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/B_Concurrent/Server$
```

(c) TCP client-server using “select”

```
Received request: Get CPU process info
Client disconnected
Received request: Get CPU process info
Client disconnected
Client disconnected
^Ctaskset: Interrupt

 Performance counter stats for 'taskset -c 1 ./server':

            129.50 msec task-clock                       #    0.002 CPUs utilized
                43      context-switches                 #  332.054 /sec
                 1      cpu-migrations                   #    7.722 /sec
               150      page-faults                      #    1.158 K/sec
      41,46,87,286      cpu_atom/cycles/                 #    3.202 GHz                         (0.27%)
      50,55,85,676      cpu_core/cycles/                 #    3.904 GHz                         (99.73%)
      35,10,79,465      cpu_atom/instructions/           #    0.85  insn per cycle              (0.27%)
      78,72,80,251      cpu_core/instructions/           #    1.90  insn per cycle              (99.73%)
       6,40,65,129      cpu_atom/branches/               #  494.723 M/sec                       (0.27%)
      14,25,94,755      cpu_core/branches/               #    1.101 G/sec                       (99.73%)
         27,31,451      cpu_atom/branch-misses/          #    4.26% of all branches             (0.27%)
          8,51,422      cpu_core/branch-misses/          #    1.33% of all branches             (99.73%)
             TopdownL1 (cpu_core)                 #     38.5 %  tma_backend_bound
                                                  #      3.6 %  tma_bad_speculation
                                                  #     28.3 %  tma_frontend_bound
                                                  #     29.6 %  tma_retiring             (99.73%)
             TopdownL1 (cpu_atom)                 #     45.3 %  tma_bad_speculation
                                                  #     21.5 %  tma_retiring             (0.27%)
                                                  #      0.0 %  tma_backend_bound
                                                  #      0.0 %  tma_backend_bound_aux
                                                  #     33.2 %  tma_frontend_bound       (0.27%)

      57.251991574 seconds time elapsed

       0.028114000 seconds user
       0.102044000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/C_Select_TCP/Server$
```

```
Server replied:
PID: 2583, Name: systemd, CPU Time: 2988611
PID: 1, Name: systemd, CPU Time: 609071

^Ctaskset: Interrupt

 Performance counter stats for 'taskset -c 0 ./client 100':

             11.40 msec task-clock                       #    0.000 CPUs utilized
               243      context-switches                 #   21.315 K/sec
                 1      cpu-migrations                   #   87.717 /sec
               337      page-faults                      #   29.561 K/sec
       1,32,63,321      cpu_atom/cycles/                 #    1.163 GHz                         (8.51%)
       3,97,80,690      cpu_core/cycles/                 #    3.489 GHz                         (96.78%)
       1,13,22,839      cpu_atom/instructions/           #    0.85  insn per cycle              (8.51%)
       2,89,84,462      cpu_core/instructions/           #    2.19  insn per cycle              (96.78%)
         20,64,269      cpu_atom/branches/               #  181.072 M/sec                       (8.51%)
         52,60,257      cpu_core/branches/               #  461.414 M/sec                       (96.78%)
            88,102      cpu_atom/branch-misses/          #    4.27% of all branches             (8.51%)
            77,990      cpu_core/branch-misses/          #    3.78% of all branches             (96.78%)
             TopdownL1 (cpu_core)                 #     30.7 %  tma_backend_bound
                                                  #      3.9 %  tma_bad_speculation
                                                  #     45.9 %  tma_frontend_bound
                                                  #     19.5 %  tma_retiring             (96.78%)
             TopdownL1 (cpu_atom)                 #     44.5 %  tma_bad_speculation
                                                  #     21.7 %  tma_retiring             (8.51%)
                                                  #      0.0 %  tma_backend_bound
                                                  #      0.0 %  tma_backend_bound_aux
                                                  #     33.8 %  tma_frontend_bound       (8.51%)

      50.231236759 seconds time elapsed

       0.000621000 seconds user
       0.007452000 seconds sys


yash@yash-Linux:~/Documents/CN_ASSIGNMENT_2/Q2/C_Select_TCP/Client$
```

Analysis:

In a single-threaded TCP server, you handle one connection at a time. The server listens for a connection, accepts it, processes it, and then moves on to the next one. Since there's no concurrency, the **number of context switches is minimal** (typically just when the kernel schedules other processes or I/O interrupts).

In a multithreaded TCP server, each incoming connection is handled in a separate thread. Each thread handles a single client, allowing multiple clients to be served concurrently. However, this introduces **more context switches** as the operating system switches between threads.

With select, the server can monitor multiple sockets at once without creating multiple threads or processes. The system call waits for any of the monitored sockets to become ready for I/O, **reducing the need for context switches compared to multithreading**, but still allowing concurrent handling of multiple connections.

Metric	Server Performance	Client Performance	Explanation
Task Clock Time	0.81 ms	0.95 ms	The task clock time for both server and client is low, indicating minimal CPU activity relative to their overall execution times.
CPU Utilization	0.000 CPUs utilized	0.535 CPUs utilized	The server shows very low CPU utilization, reflecting its idle time while waiting for client requests. The client utilizes about half of a CPU core.
Context Switches	2 context switches	2 context switches	Both the server and client have low context switch counts, suggesting minimal interruptions, which is beneficial for performance.
CPU Migrations	1 migration	1 migration	A low number of CPU migrations indicates that both processes mostly run on the same core, minimizing overhead from switching between cores.
Page Faults	127 page faults	133 page faults	Similar page fault counts indicate both processes frequently access data not currently in physical memory, potentially leading to performance overhead.
Branch Misses	17,178 branch misses (4.42% of branches)	17,864 branch misses (4.61% of branches)	Low branch miss rates for both indicate effective branch prediction, which helps maintain performance efficiency.
Cycles per Instruction	1.12 instructions per cycle	1.08 instructions per cycle	Both the server and client exhibit similar efficiency in CPU utilization, with a reasonable number of instructions executed per cycle.

Metric	Server Performance	Client Performance	Explanation
Task Clock Time	3,718.90 ms	16.04 ms	The server's task clock time is significantly higher than the client's, indicating it handles more work or multiple connections during the execution.
CPU Utilization	0.057 CPUs utilized	0.004 CPUs utilized	Both the server and client show very low CPU utilization, suggesting that they are mostly waiting for I/O operations rather than performing CPU-intensive tasks.
Context Switches	1,557 context switches	442 context switches	The server has a higher context switch count, which is expected in a multithreaded environment as it manages multiple client connections.
CPU Migrations	631 migrations	9 migrations	The server experiences more CPU migrations, indicating it may be handling tasks on different CPU cores, which can lead to increased overhead.
Page Faults	169 page faults	62 page faults	The server has a higher count of page faults, suggesting it accesses more data not currently in physical memory, potentially affecting performance.
Branch Misses	2,57,01,827 branch misses (4.41% of branches)	58,539 branch misses (4.31% of branches)	The server has a higher count of branch misses, which could affect its performance but is still within acceptable limits for branch prediction efficiency.
TopdownL1 (cpu_core)	39.5% frontend bound	54.0% frontend bound	A significant percentage of cycles for both processes is spent on handling frontend operations, indicating time spent waiting for memory and data retrieval.

Metric	Server Performance	Client Performance	Explanation
Task Clock Time	129.50 ms	11.40 ms	The server's task clock time is notably higher than the client's, indicating it handles more connections and processing during execution.
CPU Utilization	0.002 CPUs utilized	0.000 CPUs utilized	Both processes show minimal CPU utilization, suggesting they primarily wait for I/O operations rather than executing CPU-intensive tasks.
Context Switches	150 context switches	337 context switches	The client has more context switches, indicating frequent task switching, which may be due to multiple requests being processed concurrently.
CPU Migrations	43 migrations	1 migration	The server has a higher number of CPU migrations, which is typical in a multithreaded environment, allowing better distribution of tasks across CPU cores.
Page Faults	332 page faults	21 page faults	The server experiences more page faults, which may indicate it accesses more data that isn't in physical memory, potentially impacting performance.
Branch Misses	1,32,63,321 branch misses (4.27% of branches)	88,102 branch misses (4.4% of branches)	Both the server and client maintain an efficient branch prediction, with a relatively low percentage of branch misses.
TopdownL1 (cpu_core)	45.9% frontend bound	19.5% frontend bound	A significant percentage of cycles for the server is spent on handling frontend operations, indicating time spent waiting for memory and data retrieval.