[Multithreaded Server]

* C++ Multithreaded Server

```cpp
#include <winsock2.h>
#include <ws2tcpip.h>
#include <iostream>
#include <string>
#include <thread>
#include <vector>
#include <mutex>

// Link with ws2_32.lib
#pragma comment(lib, "ws2_32.lib")

const int PORT = 8080;
std::mutex cout_mutex; // To synchronize console output

// Function to handle a client connection
void handle_client(SOCKET client_socket) {
    char buffer[1024] = { 0 };
    const std::string response =
        "HTTP/1.1 200 OK\r\n"
        "Content-Type: text/plain\r\n"
        "Content-Length: 13\r\n"
        "\r\n"
        "Hello, World!";

    // Read client request (for demonstration, the request is ignored)
    recv(client_socket, buffer, sizeof(buffer), 0);

    // Respond to the client
    send(client_socket, response.c_str(), response.size(), 0);

    // Thread-safe console output
    {
        std::lock_guard<std::mutex> lock(cout_mutex);
        std::cout << "Responded to client\n";
    }

    // Close the client socket
    closesocket(client_socket);
}
```

per thread execution

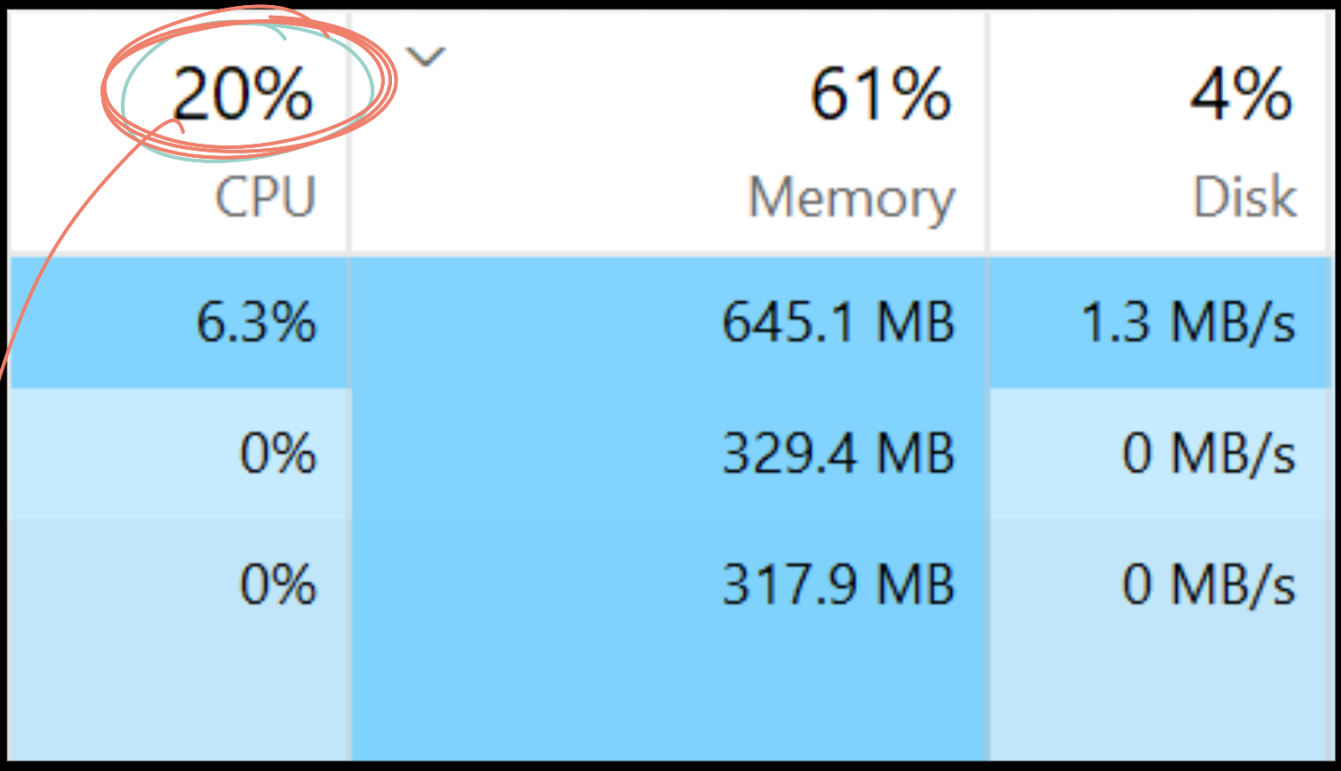
locking

```cpp
int main() {
    WSADATA wsaData;
    SOCKET server_fd = INVALID_SOCKET;
    struct sockaddr_in address;
    int addrlen = sizeof(address);

    // Initialize Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        std::cerr << "WSAStartup failed: " << WSAGetLastError() << "\n";
        return -1;
    }

    // Create socket
    if ((server_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == INVALID_SOCKET) {
        std::cerr << "Socket creation failed: " << WSAGetLastError() << "\n";
        WSACleanup();
        return -1;
    }

    // Bind socket to port
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);
    if (bind(server_fd, (struct sockaddr*)&address, sizeof(address)) == SOCKET_ERROR) {
        std::cerr << "Bind failed: " << WSAGetLastError() << "\n";
        closesocket(server_fd);
        WSACleanup();
        return -1;
    }

    // Listen for incoming connections
    if (listen(server_fd, 3) == SOCKET_ERROR) {
        std::cerr << "Listen failed: " << WSAGetLastError() << "\n";
        closesocket(server_fd);
        WSACleanup();
        return -1;
    }

    std::cout << "Server is listening on port " << PORT << "...\n";
```

```cpp
// Vector to hold threads for managing client connections
std::vector<std::thread> threads;

while (true) {
    std::cout << "Inside While Loop\n";
    SOCKET client_socket;
    if ((client_socket = accept(server_fd, (struct sockaddr*)&address, &addrlen)) == INVALID_SOCKET) {
        std::cerr << "Accept failed: " << WSAGetLastError() << "\n";
        continue;
    }

    // Thread-safe console output
    {
        std::lock_guard<std::mutex> lock(cout_mutex);
        std::cout << "New client connected\n";
    }

    // Launch a thread to handle the client
    threads.emplace_back(handle_client, client_socket);
}

// Cleanup
for (auto& t : threads) {
    if (t.joinable()) {
        t.join();
    }
}

closesocket(server_fd);
WSACleanup();
return 0;
}
```

threads

threads (concurrency or could be parallelism if Multiple core



20% CPU	61% Memory	4% Disk
6.3%	645.1 MB	1.3 MB/s
0%	329.4 MB	0 MB/s
0%	317.9 MB	0 MB/s

→ Multithread server increase CPU throughput as it utilizes it using Multiple concurrent thread

& Memory usage is not Bloat up because as the req is recieved, it gets processed.

Problems

1. Thread accessing some Data
   - Need synchronization
2. Application code is hard to write