Marmara University
Faculty of Engineering

CSE 4074
Computer Networks

Programming Assignment

Instructor: Ömer Korçak

Date:29.12.2024

	Department	Student Id Number	Name & Surname
1	CSE	150120055	Muhammed Talha KARAGÜL
2	CSE	150121021	Feyzullah Asıllıoğlu
3	CSE	150121076	Abdullah KAN

How to Run The Program

To run the program we'll be needed 3 terminals. In the programs directory run each terminal. In the first terminal run

→ "java server.java <server port number>",

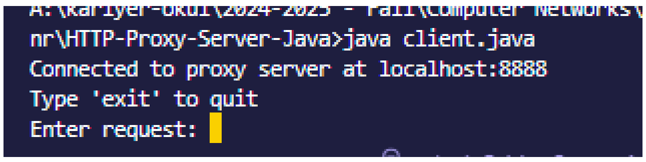
In the second terminal run

→ " java proxy.java <server port number>",

In the third terminal run

→ "java client.java".

After running client.java, the user should enter the intended request to the program:



```
A:\kariyer-okul\2024-2025 - Fall\Computer Networks\
nr\HTTP-Proxy-Server-Java>java client.java
Connected to proxy server at localhost:8888
Type 'exit' to quit
Enter request: 
```

The request can be these three:

→ Single URI (e.g. 5000)

→ Relative Path: GET /500 HTTP/1.1

→ Full URI: GET http://localhost:8080/500 HTTP/1.1

After the request was made, the file that was received from the server was saved to the file if input to the program was appropriate.

Abstract

The aim of this program is to enable a multi-threaded HTTP server and proxy server to run attached, with special attention to socket programming and concurrency management. The HTTP server can process GET requests, generates HTML pages dynamically based on a prescribed size, and is standard-compliant in terms of the HTTP protocol. Invalid requests will generate suitable error conditions, and it will also allow client connections through a browser.

In addition to this, the proxy server works in association with this HTTP server to forward the client request and respond to some other constraints, including URI length and browser proxy support. This project will optionally implement cache mechanisms.

The performance evaluation, in this case, is done on both servers using the tool called "wrk". The parameters compared will be latency and throughput under various levels of concurrency and connections. This report discusses design, implementation, and testing methodologies, revealing server behavior under load and possible areas for further optimization.

Implementation

Server:

- **Request Parsing:**
    - Extract the method, URI, and protocol version from incoming requests.
    - Validate that the method is GET and the size parameter is within the permissible range.
- **Content Generation:**
    - Dynamically generate HTML documents to meet requested sizes using structured prefixes and suffixes.
- **Response Formatting:**
    - Construct HTTP-compliant responses with headers (Content-Type, Content-Length) and the generated content.
- **Error Handling:**
    - Implement custom exceptions to handle invalid requests (e.g., unsupported HTTP methods or invalid sizes).

Proxy

- **Connection Management:**
    - Establish a listening socket to accept client connections.
    - Use multithreading to handle multiple clients concurrently.
- **Request Validation:**
    - Parse and validate incoming client requests to ensure they meet predefined constraints (e.g., URI format, maximum size).
- **Request Forwarding:**
    - Forward valid requests to the server after rewriting the URI to a relative format.
- **Response Relaying:**
    - Stream server responses directly back to the client, maintaining the original content.
- **Error Responses:**
    - Generate and return HTTP error messages for invalid requests or server unavailability.

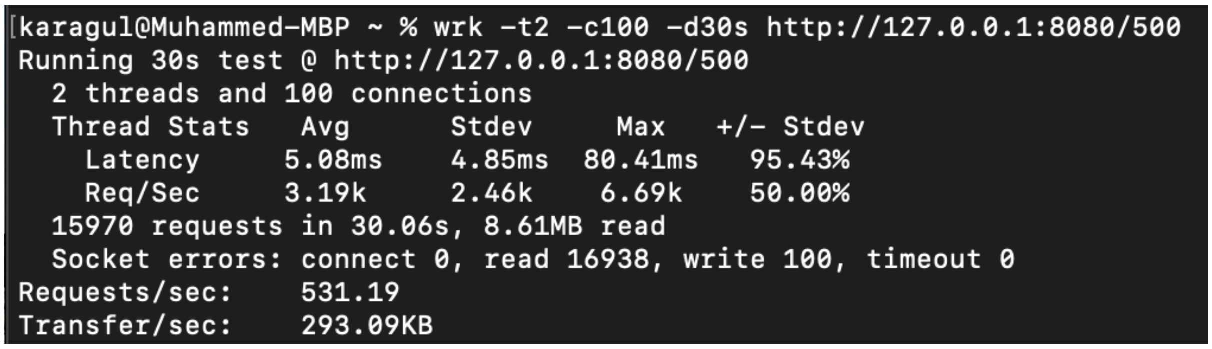
Client

- **User Interaction:**
    - Prompt the user to input HTTP requests.
    - Display server responses and save them locally as HTML files.
- **Request Formatting:**
    - Construct valid HTTP requests dynamically, ensuring compatibility with server and proxy requirements.
- **Response Processing:**
    - Parse server responses to extract headers, status codes, and content.
    - Display the parsed content or errors to the user.

Performance Testing for Web Server

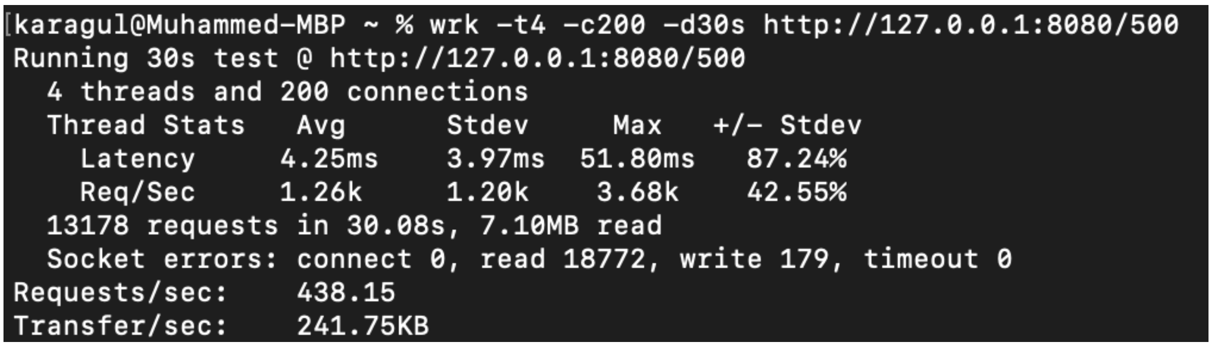
Effect of Increasing Threads and Connections on Performance

Test 1:



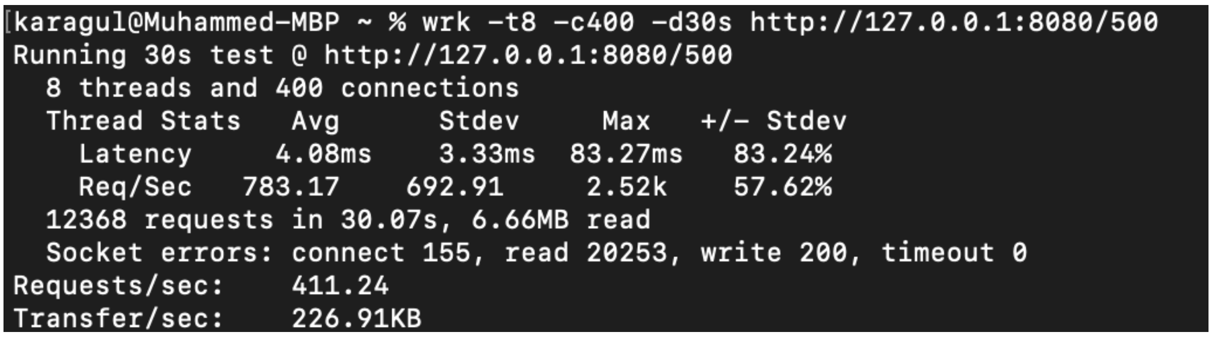
```
[karagul@Muhammed-MBP ~ % wrk -t2 -c100 -d30s http://127.0.0.1:8080/500
 Running 30s test @ http://127.0.0.1:8080/500
   2 threads and 100 connections
   Thread Stats   Avg      Stdev     Max   +/- Stdev
     Latency     5.08ms    4.85ms  80.41ms   95.43%
     Req/Sec     3.19k     2.46k    6.69k    50.00%
   15970 requests in 30.06s, 8.61MB read
   Socket errors: connect 0, read 16938, write 100, timeout 0
 Requests/sec:    531.19
 Transfer/sec:    293.09KB
```

Test 2:



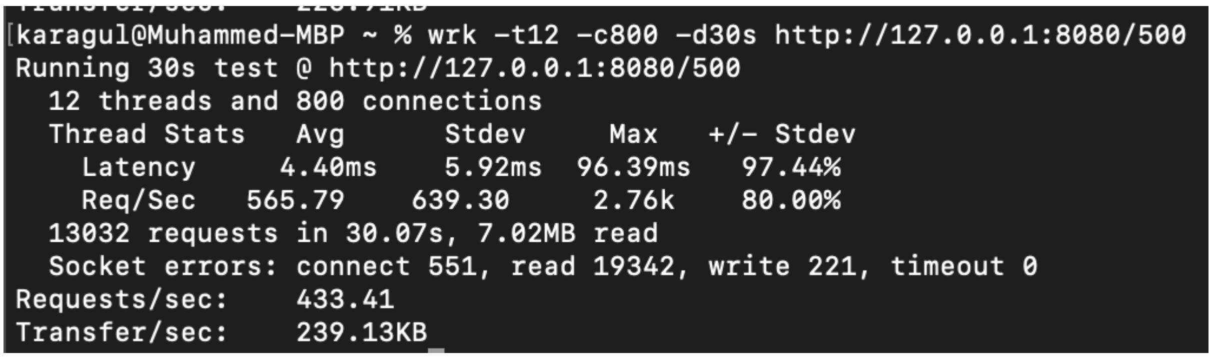
```
[karagul@Muhammed-MBP ~ % wrk -t4 -c200 -d30s http://127.0.0.1:8080/500
 Running 30s test @ http://127.0.0.1:8080/500
   4 threads and 200 connections
   Thread Stats   Avg      Stdev     Max   +/- Stdev
     Latency     4.25ms    3.97ms  51.80ms   87.24%
     Req/Sec     1.26k     1.20k    3.68k    42.55%
   13178 requests in 30.08s, 7.10MB read
   Socket errors: connect 0, read 18772, write 179, timeout 0
 Requests/sec:    438.15
 Transfer/sec:    241.75KB
```

Test 3:



```
[karagul@Muhammed-MBP ~ % wrk -t8 -c400 -d30s http://127.0.0.1:8080/500
 Running 30s test @ http://127.0.0.1:8080/500
   8 threads and 400 connections
   Thread Stats   Avg      Stdev     Max   +/- Stdev
     Latency     4.08ms    3.33ms  83.27ms   83.24%
     Req/Sec   783.17    692.91     2.52k    57.62%
   12368 requests in 30.07s, 6.66MB read
   Socket errors: connect 155, read 20253, write 200, timeout 0
 Requests/sec:    411.24
 Transfer/sec:    226.91KB
```

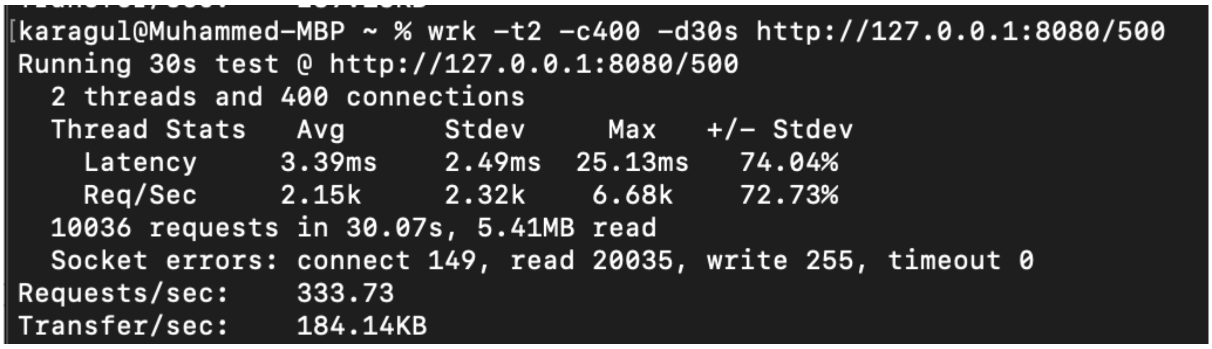
Test 4:



```
[karagul@Muhammed-MBP ~ % wrk -t12 -c800 -d30s http://127.0.0.1:8080/500
 Running 30s test @ http://127.0.0.1:8080/500
   12 threads and 800 connections
   Thread Stats   Avg      Stdev     Max   +/- Stdev
     Latency     4.40ms    5.92ms  96.39ms   97.44%
     Req/Sec   565.79    639.30     2.76k    80.00%
   13032 requests in 30.07s, 7.02MB read
   Socket errors: connect 551, read 19342, write 221, timeout 0
 Requests/sec:    433.41
 Transfer/sec:    239.13KB
```

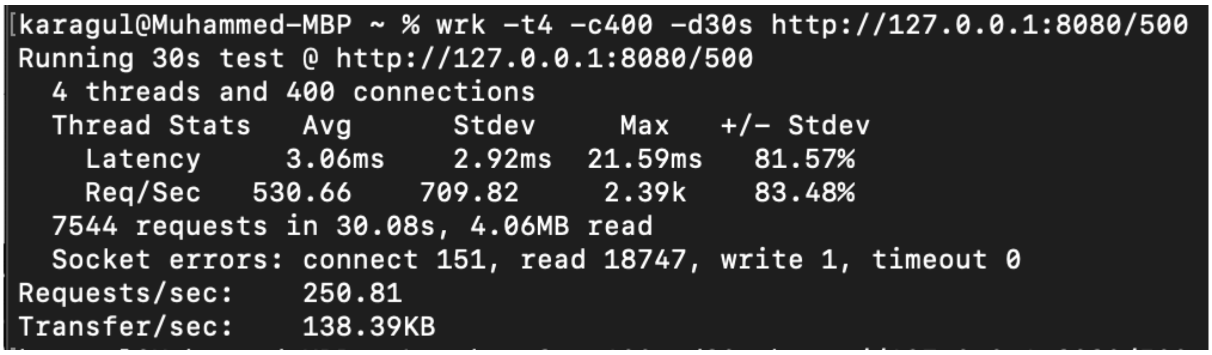
Effect of Increasing Threads on Performance

Test 5:



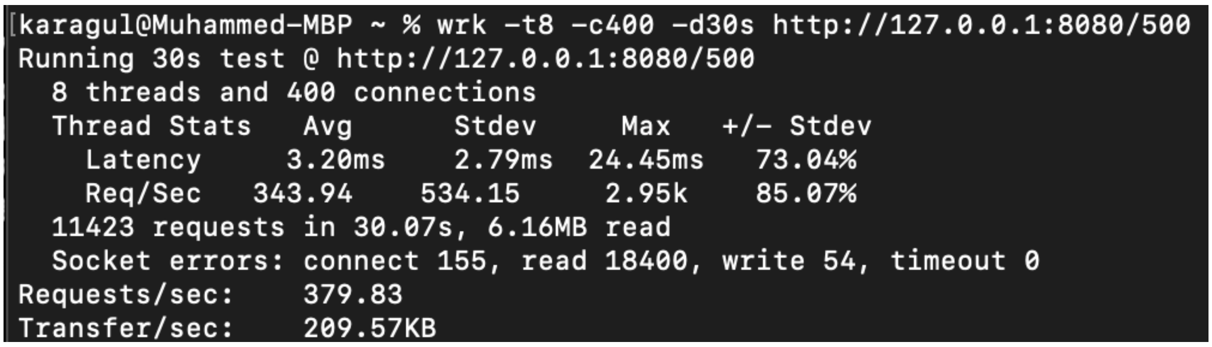
```
karagul@Muhammed-MBP ~ % wrk -t2 -c400 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  2 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     3.39ms    2.49ms  25.13ms   74.04%
    Req/Sec     2.15k     2.32k    6.68k    72.73%
  10036 requests in 30.07s, 5.41MB read
  Socket errors: connect 149, read 20035, write 255, timeout 0
Requests/sec:    333.73
Transfer/sec:    184.14KB
```

Test 6:



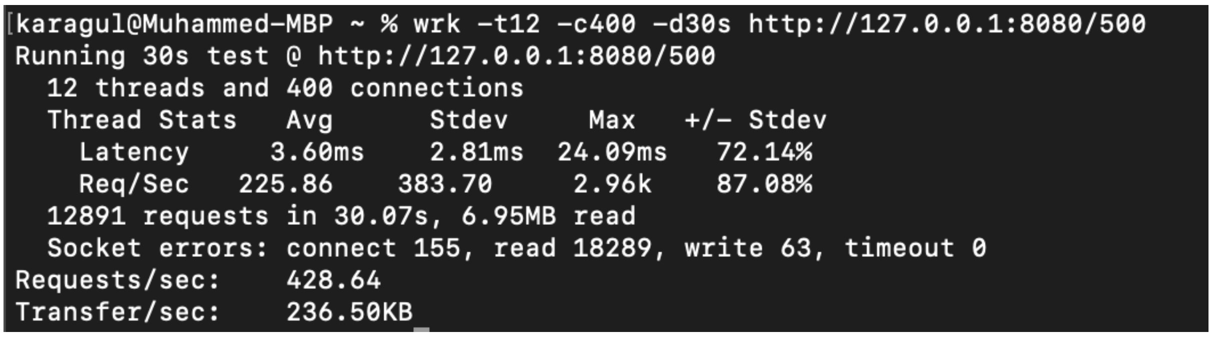
```
karagul@Muhammed-MBP ~ % wrk -t4 -c400 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  4 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     3.06ms    2.92ms  21.59ms   81.57%
    Req/Sec   530.66    709.82     2.39k    83.48%
  7544 requests in 30.08s, 4.06MB read
  Socket errors: connect 151, read 18747, write 1, timeout 0
Requests/sec:    250.81
Transfer/sec:    138.39KB
```

Test 7:



```
karagul@Muhammed-MBP ~ % wrk -t8 -c400 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  8 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     3.20ms    2.79ms  24.45ms   73.04%
    Req/Sec   343.94    534.15     2.95k    85.07%
  11423 requests in 30.07s, 6.16MB read
  Socket errors: connect 155, read 18400, write 54, timeout 0
Requests/sec:    379.83
Transfer/sec:    209.57KB
```

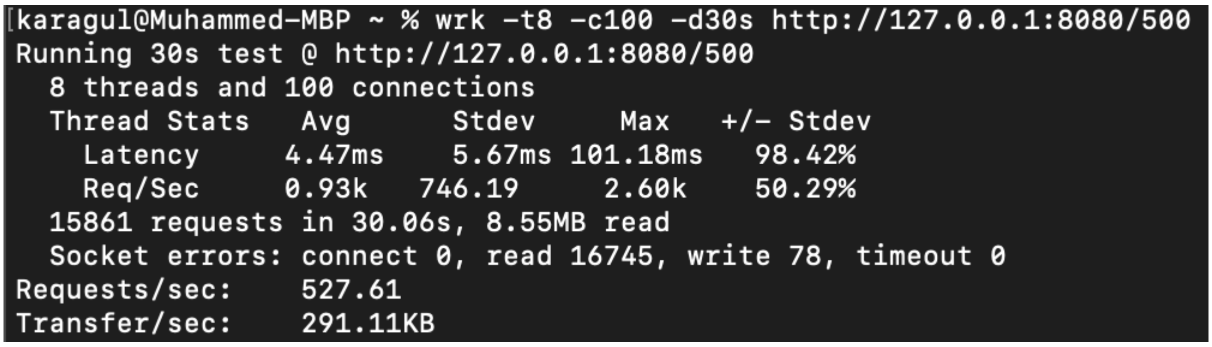
Test 8:



```
karagul@Muhammed-MBP ~ % wrk -t12 -c400 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  12 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     3.60ms    2.81ms  24.09ms   72.14%
    Req/Sec   225.86    383.70     2.96k    87.08%
  12891 requests in 30.07s, 6.95MB read
  Socket errors: connect 155, read 18289, write 63, timeout 0
Requests/sec:    428.64
Transfer/sec:    236.50KB
```

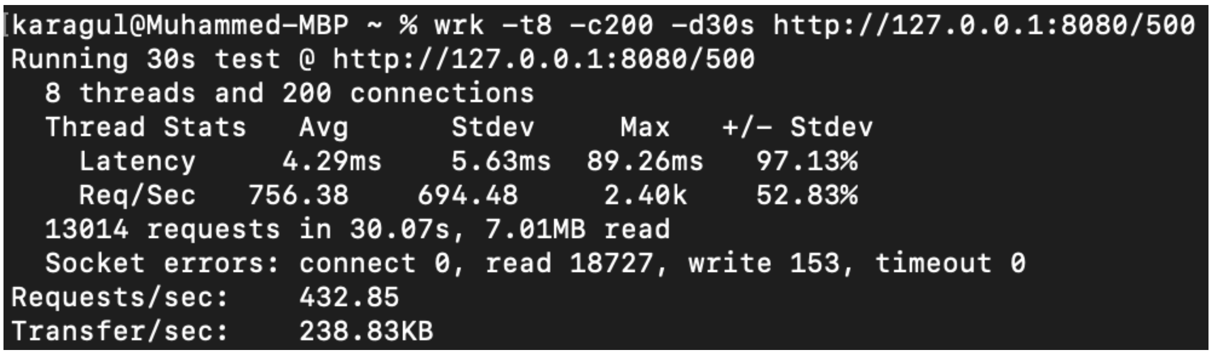
Effect of Increasing Connections on Performance

Test 9:



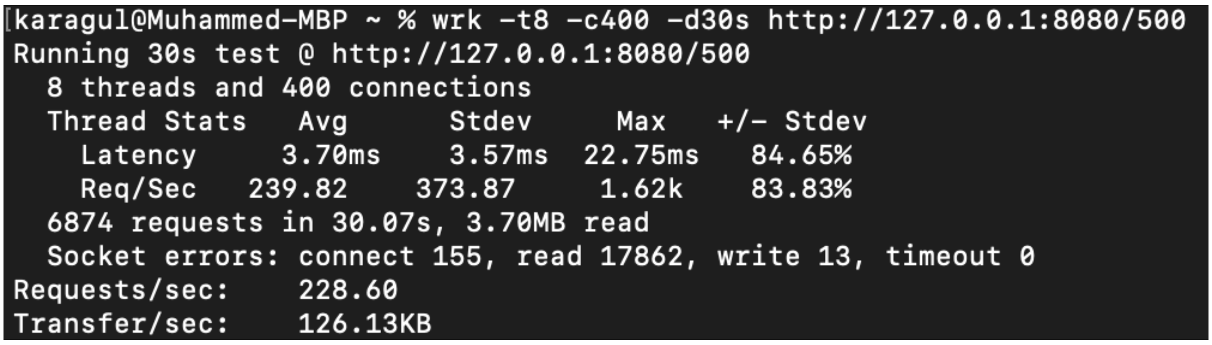
```
[karagul@Muhammed-MBP ~ % wrk -t8 -c100 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  8 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     4.47ms    5.67ms 101.18ms   98.42%
    Req/Sec     0.93k   746.19     2.60k    50.29%
  15861 requests in 30.06s, 8.55MB read
  Socket errors: connect 0, read 16745, write 78, timeout 0
Requests/sec:    527.61
Transfer/sec:    291.11KB
```

Test 10:



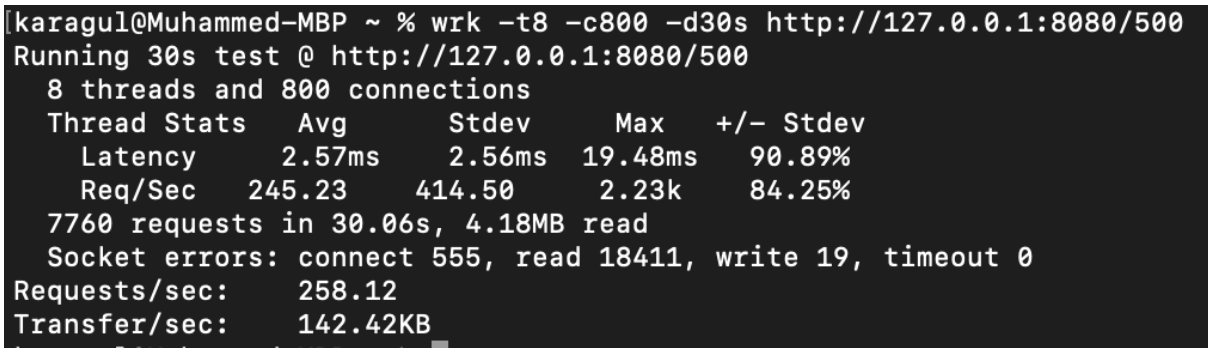
```
[karagul@Muhammed-MBP ~ % wrk -t8 -c200 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  8 threads and 200 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     4.29ms    5.63ms  89.26ms   97.13%
    Req/Sec   756.38    694.48     2.40k    52.83%
  13014 requests in 30.07s, 7.01MB read
  Socket errors: connect 0, read 18727, write 153, timeout 0
Requests/sec:    432.85
Transfer/sec:    238.83KB
```

Test 11:



```
[karagul@Muhammed-MBP ~ % wrk -t8 -c400 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  8 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     3.70ms    3.57ms  22.75ms   84.65%
    Req/Sec   239.82    373.87     1.62k    83.83%
  6874 requests in 30.07s, 3.70MB read
  Socket errors: connect 155, read 17862, write 13, timeout 0
Requests/sec:    228.60
Transfer/sec:    126.13KB
```

Test 12:

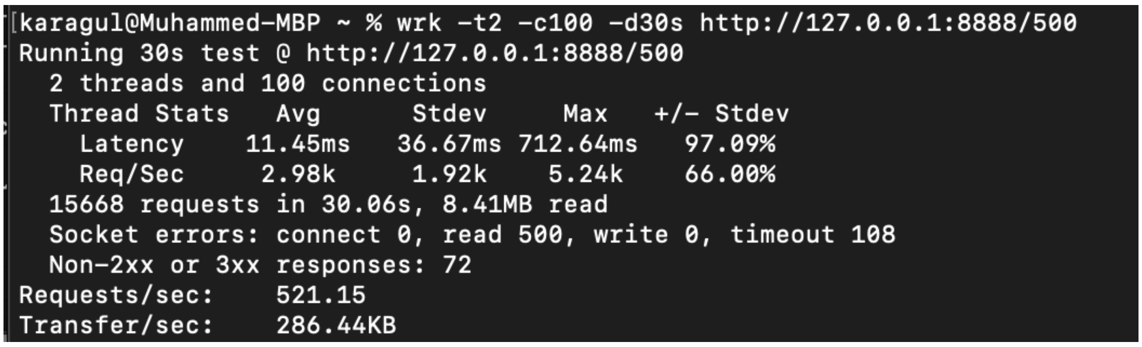


```
[karagul@Muhammed-MBP ~ % wrk -t8 -c800 -d30s http://127.0.0.1:8080/500
Running 30s test @ http://127.0.0.1:8080/500
  8 threads and 800 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     2.57ms    2.56ms  19.48ms   90.89%
    Req/Sec   245.23    414.50     2.23k    84.25%
  7760 requests in 30.06s, 4.18MB read
  Socket errors: connect 555, read 18411, write 19, timeout 0
Requests/sec:    258.12
Transfer/sec:    142.42KB
```

Performance Testing for Proxy Server

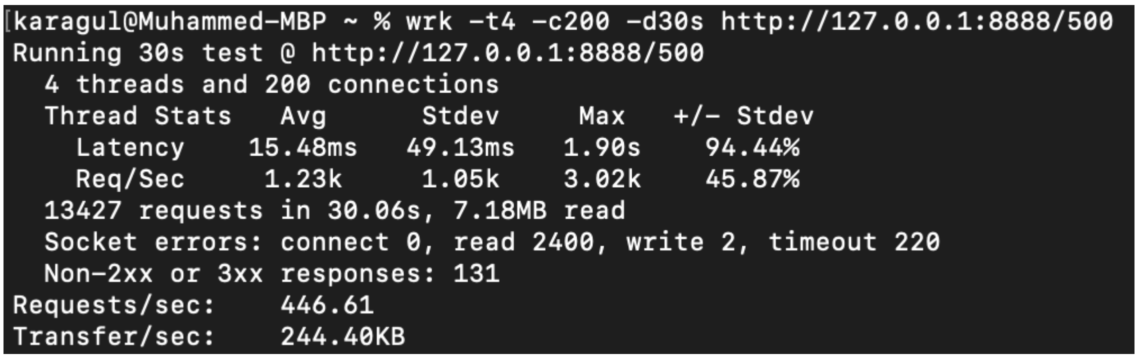
Effect of Increasing Threads and Connections on Performance

Test 1:



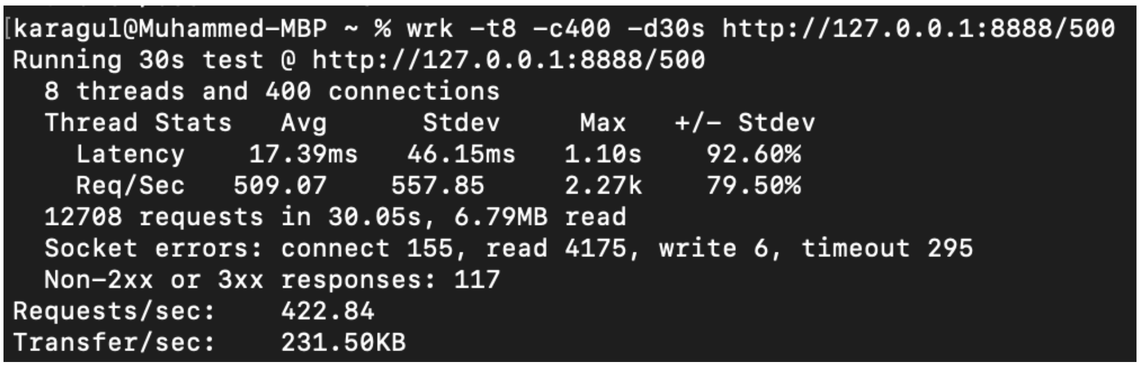
```
karagul@Muhammed-MBP ~ % wrk -t2 -c100 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  2 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    11.45ms   36.67ms 712.64ms   97.09%
    Req/Sec     2.98k     1.92k    5.24k    66.00%
  15668 requests in 30.06s, 8.41MB read
  Socket errors: connect 0, read 500, write 0, timeout 108
  Non-2xx or 3xx responses: 72
Requests/sec:    521.15
Transfer/sec:    286.44KB
```

Test 2:



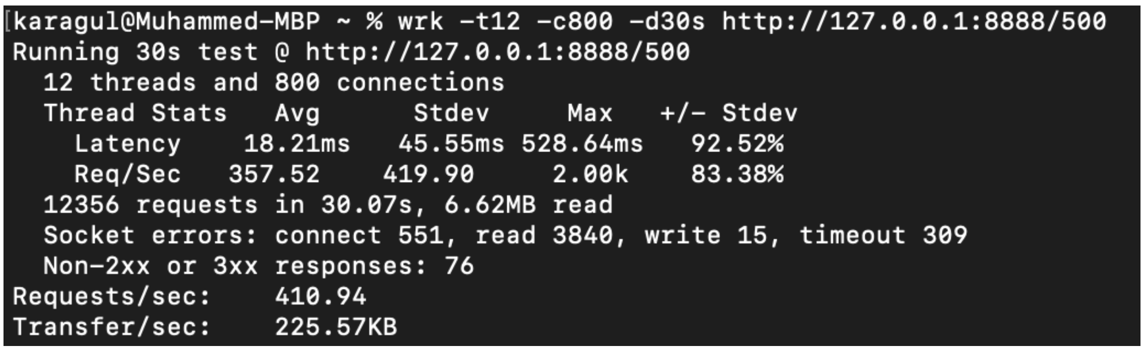
```
karagul@Muhammed-MBP ~ % wrk -t4 -c200 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  4 threads and 200 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    15.48ms   49.13ms   1.90s    94.44%
    Req/Sec     1.23k     1.05k    3.02k    45.87%
  13427 requests in 30.06s, 7.18MB read
  Socket errors: connect 0, read 2400, write 2, timeout 220
  Non-2xx or 3xx responses: 131
Requests/sec:    446.61
Transfer/sec:    244.40KB
```

Test 3:



```
karagul@Muhammed-MBP ~ % wrk -t8 -c400 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  8 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    17.39ms   46.15ms   1.10s    92.60%
    Req/Sec   509.07    557.85     2.27k    79.50%
  12708 requests in 30.05s, 6.79MB read
  Socket errors: connect 155, read 4175, write 6, timeout 295
  Non-2xx or 3xx responses: 117
Requests/sec:    422.84
Transfer/sec:    231.50KB
```

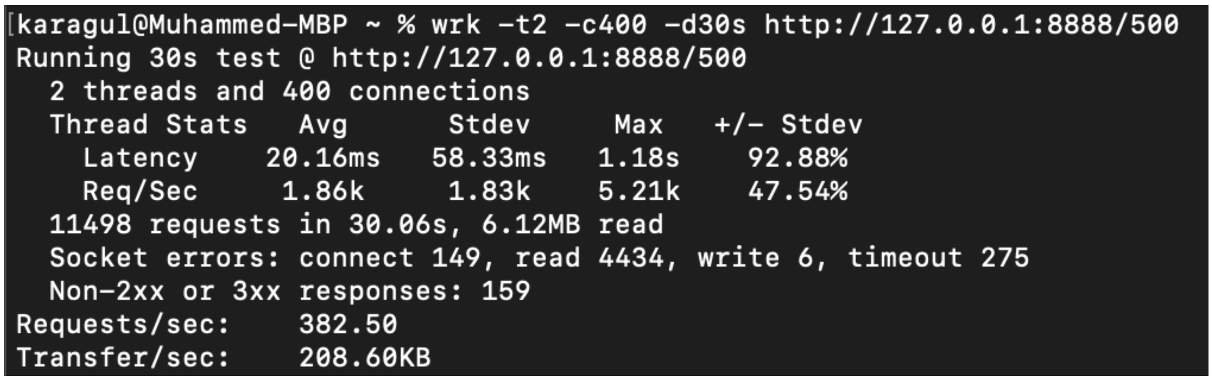
Test 4:



```
karagul@Muhammed-MBP ~ % wrk -t12 -c800 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  12 threads and 800 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    18.21ms   45.55ms 528.64ms   92.52%
    Req/Sec   357.52    419.90     2.00k    83.38%
  12356 requests in 30.07s, 6.62MB read
  Socket errors: connect 551, read 3840, write 15, timeout 309
  Non-2xx or 3xx responses: 76
Requests/sec:    410.94
Transfer/sec:    225.57KB
```

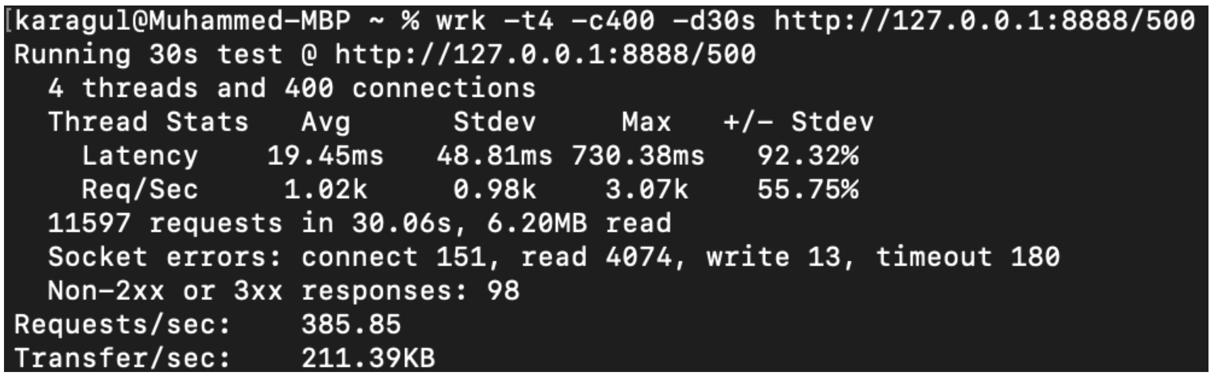
Effect of Increasing Threads on Performance

Test 5:



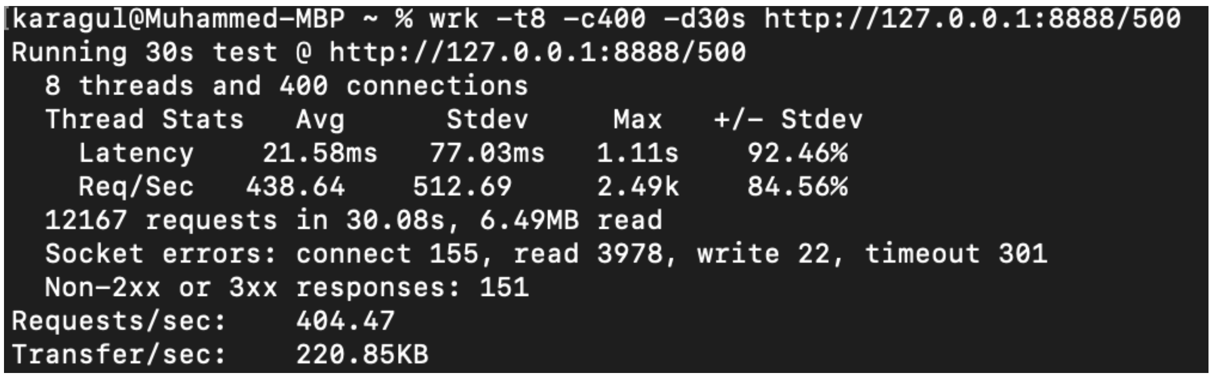
```
karagul@Muhammed-MBP ~ % wrk -t2 -c400 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  2 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    20.16ms   58.33ms   1.18s    92.88%
    Req/Sec     1.86k     1.83k    5.21k    47.54%
  11498 requests in 30.06s, 6.12MB read
  Socket errors: connect 149, read 4434, write 6, timeout 275
  Non-2xx or 3xx responses: 159
Requests/sec:    382.50
Transfer/sec:    208.60KB
```

Test 6:



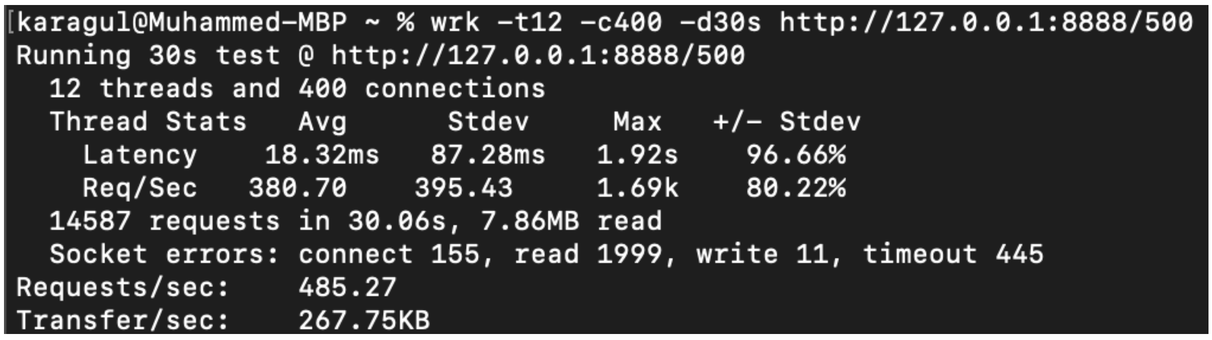
```
karagul@Muhammed-MBP ~ % wrk -t4 -c400 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  4 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    19.45ms   48.81ms 730.38ms   92.32%
    Req/Sec     1.02k     0.98k    3.07k    55.75%
  11597 requests in 30.06s, 6.20MB read
  Socket errors: connect 151, read 4074, write 13, timeout 180
  Non-2xx or 3xx responses: 98
Requests/sec:    385.85
Transfer/sec:    211.39KB
```

Test 7:



```
karagul@Muhammed-MBP ~ % wrk -t8 -c400 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  8 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    21.58ms   77.03ms   1.11s    92.46%
    Req/Sec   438.64    512.69     2.49k    84.56%
  12167 requests in 30.08s, 6.49MB read
  Socket errors: connect 155, read 3978, write 22, timeout 301
  Non-2xx or 3xx responses: 151
Requests/sec:    404.47
Transfer/sec:    220.85KB
```

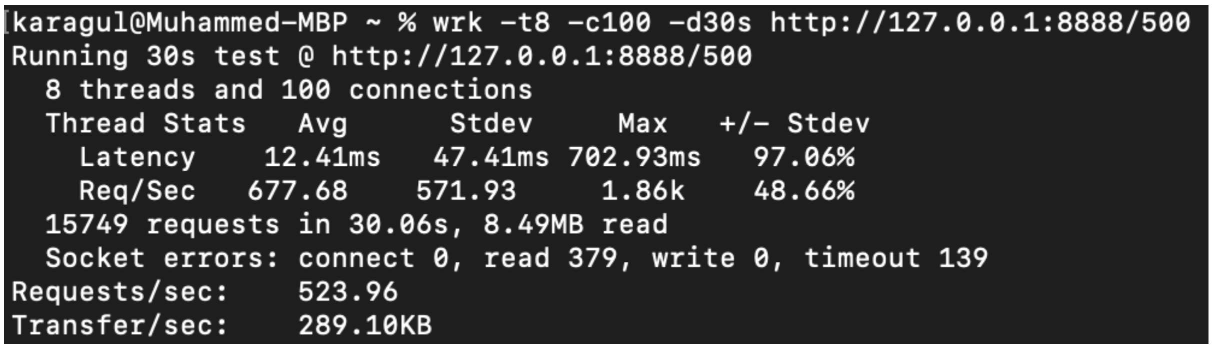
Test 8:



```
karagul@Muhammed-MBP ~ % wrk -t12 -c400 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  12 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    18.32ms   87.28ms   1.92s    96.66%
    Req/Sec   380.70    395.43     1.69k    80.22%
  14587 requests in 30.06s, 7.86MB read
  Socket errors: connect 155, read 1999, write 11, timeout 445
Requests/sec:    485.27
Transfer/sec:    267.75KB
```

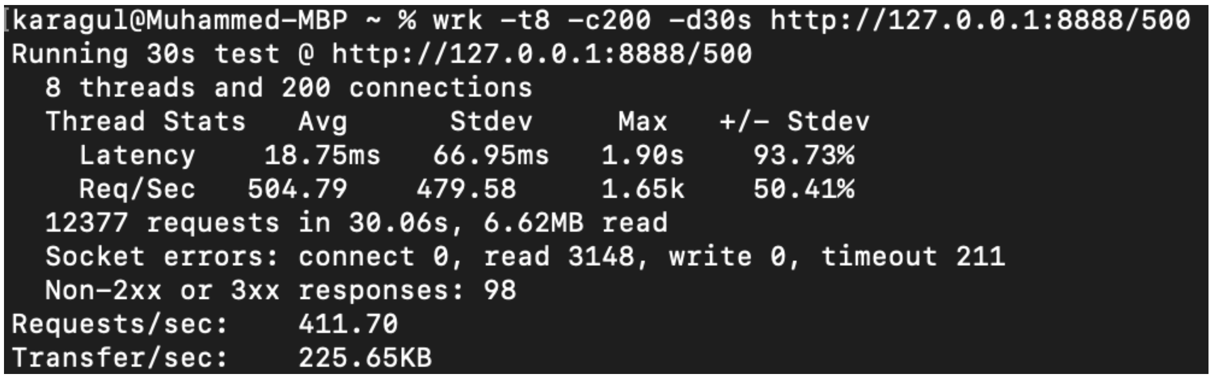
Effect of Increasing Connections on Performance

Test 9:



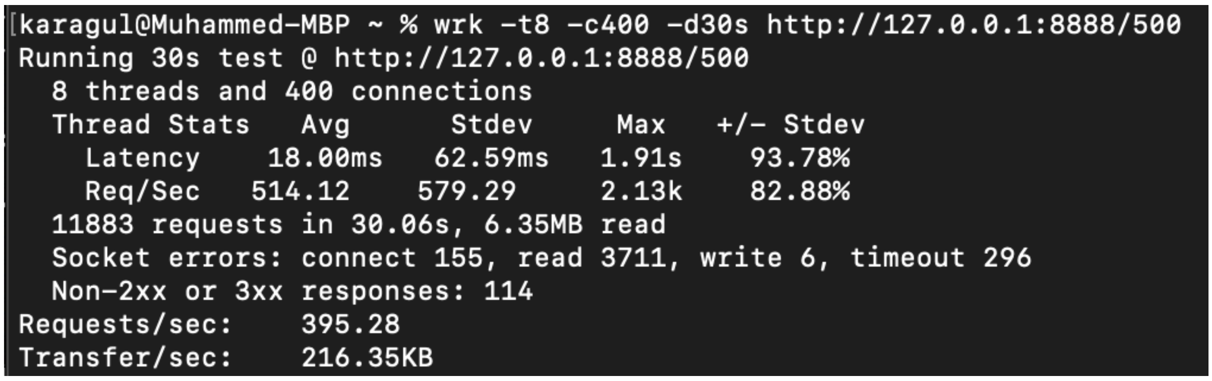
```
[karagul@Muhammed-MBP ~ % wrk -t8 -c100 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  8 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    12.41ms   47.41ms 702.93ms   97.06%
    Req/Sec   677.68    571.93     1.86k    48.66%
  15749 requests in 30.06s, 8.49MB read
  Socket errors: connect 0, read 379, write 0, timeout 139
Requests/sec:    523.96
Transfer/sec:    289.10KB
```

Test 10:



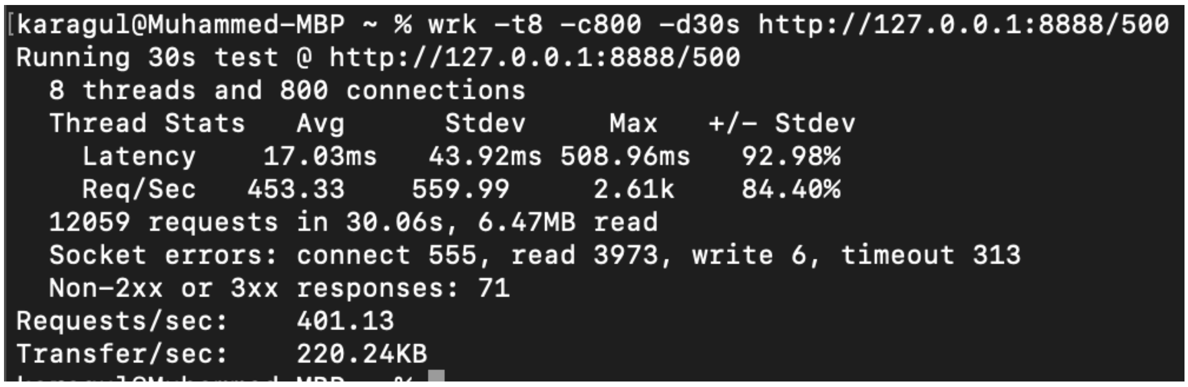
```
[karagul@Muhammed-MBP ~ % wrk -t8 -c200 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  8 threads and 200 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    18.75ms   66.95ms   1.90s    93.73%
    Req/Sec   504.79    479.58     1.65k    50.41%
  12377 requests in 30.06s, 6.62MB read
  Socket errors: connect 0, read 3148, write 0, timeout 211
  Non-2xx or 3xx responses: 98
Requests/sec:    411.70
Transfer/sec:    225.65KB
```

Test 11:



```
[karagul@Muhammed-MBP ~ % wrk -t8 -c400 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  8 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    18.00ms   62.59ms   1.91s    93.78%
    Req/Sec   514.12    579.29     2.13k    82.88%
  11883 requests in 30.06s, 6.35MB read
  Socket errors: connect 155, read 3711, write 6, timeout 296
  Non-2xx or 3xx responses: 114
Requests/sec:    395.28
Transfer/sec:    216.35KB
```

Test 12:



```
[karagul@Muhammed-MBP ~ % wrk -t8 -c800 -d30s http://127.0.0.1:8888/500
Running 30s test @ http://127.0.0.1:8888/500
  8 threads and 800 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    17.03ms   43.92ms 508.96ms   92.98%
    Req/Sec   453.33    559.99     2.61k    84.40%
  12059 requests in 30.06s, 6.47MB read
  Socket errors: connect 555, read 3973, write 6, timeout 313
  Non-2xx or 3xx responses: 71
Requests/sec:    401.13
Transfer/sec:    220.24KB
```

Tests & Conclusions

Web Server Tests

- How your server's performance changes by increasing the level of concurrency:

The server's performance demonstrates a clear trend when concurrency levels increase. In the first case, where both thread count and connection count increased together, the requests per second (RPS) generally decreased. This indicates that the server struggled to handle higher levels of simultaneous requests efficiently, likely due to resource contention or limitations in processing capacity. Similarly, in the second case, where thread count increased while connection count remained stable, performance was more consistent, but marginal improvements were seen with additional threads. This suggests the server benefits from parallelism to a point, but the returns diminish with higher thread counts due to overhead or synchronization costs. In the third case, where threads were stable, but connection counts increased, the RPS dropped significantly as the server had to manage more concurrent connections. This highlights the bottleneck introduced by excessive connections, emphasizing that optimal performance requires a balanced concurrency level tailored to the server's resources.

- Maximum of how many concurrent requests your server handles in a reasonable time:

The server maintained reasonable response times and throughput at lower concurrency levels, such as with two threads and 100 connections, achieving an RPS of 531.19. However, as concurrency increased—either through additional threads or higher connection counts—the performance degraded, as seen in cases with 8 threads and 400 or 800 connections. Here, the RPS dropped below 300, and socket errors, such as failed reads, became more prominent. Therefore, the server appears capable of handling around 100-200 concurrent requests with 2-4 threads efficiently while maintaining acceptable latency and minimizing errors. Beyond this threshold, the server's performance declines, indicating resource constraints or architectural limitations that need addressing to scale further.

Proxy Server Tests

How Your Server’s Performance Changes by Increasing Level of Concurrency

The server's performance exhibited noticeable variations as concurrency levels increased, depending on the configuration of threads and connections. When both threads and connections were increased simultaneously, the performance initially improved, with higher requests per second and throughput. However, as concurrency levels grew, the server's performance began to degrade, showing higher latency and increasing socket errors such as timeouts and read errors. This indicates that the server's ability to handle simultaneous requests is constrained by resource limitations, such as CPU, memory, or I/O handling capacity.

In scenarios where the number of threads increased while connections remained stable, there was a slight improvement in handling latency, and the server could process more requests per second. However, this effect plateaued beyond a certain point, as adding more threads without increasing connections had diminishing returns. This suggests the server's thread management and parallel processing capabilities have a limit, beyond which performance gains become negligible.

When connections increased while threads remained stable, the server faced a greater challenge. Latency increased significantly, and the number of requests processed dropped, coupled with a rise in socket errors. This scenario emphasizes the importance of balancing threads and connections for optimal performance since a disproportionate increase in connections overwhelms the server's processing capabilities.

Maximum of How Many Concurrent Requests Your Server Can Handle in a Reasonable Time

Based on the test outputs, the server was able to handle around 400 connections with 8 threads in a relatively reasonable time, maintaining throughput of approximately 395-422 requests per second with acceptable latency. Beyond this point, increasing concurrency led to more socket errors, higher latencies, and reduced stability, indicating the server's upper limit for concurrent request handling under the given conditions. Further optimization in the server configuration or resource allocation might help push this boundary.