Assignment 1
University of Cyprus
CS425

Stepan Zalis
Loukas Soleas

TCP Client

This class simulates of 10 TCP clients who establish a connection with a server hosted on a virtual machine and requesting data from him. It is implemented using multithreading. One of our biggest consideration was to decide to use ExecutorService (thread pool) or another solution which could be reactive implementation of Java - RxJava. For simple application like our it was easier to create ExecuterService, but for future development we would use RxJava, because it has more advanced technique how to handle asynchronous. Every TCP client is a thread and clients are running concurrently. Every client generates a series of requests towards to the server.The client sends a hello message which comprises of a “Hello" string the client's ip and the port and the client id.

TCP Server

Server program waits for a number of clients to connect. Once the server receives a message from client, it as soon as he can generates a response message and sends it back to the client . The response message comprises of a "Welcome” $clientID and a variable with randomly sized generated payload (for simulation we decided to use a String “a” and its size should be between 300 kb and 2 mb.

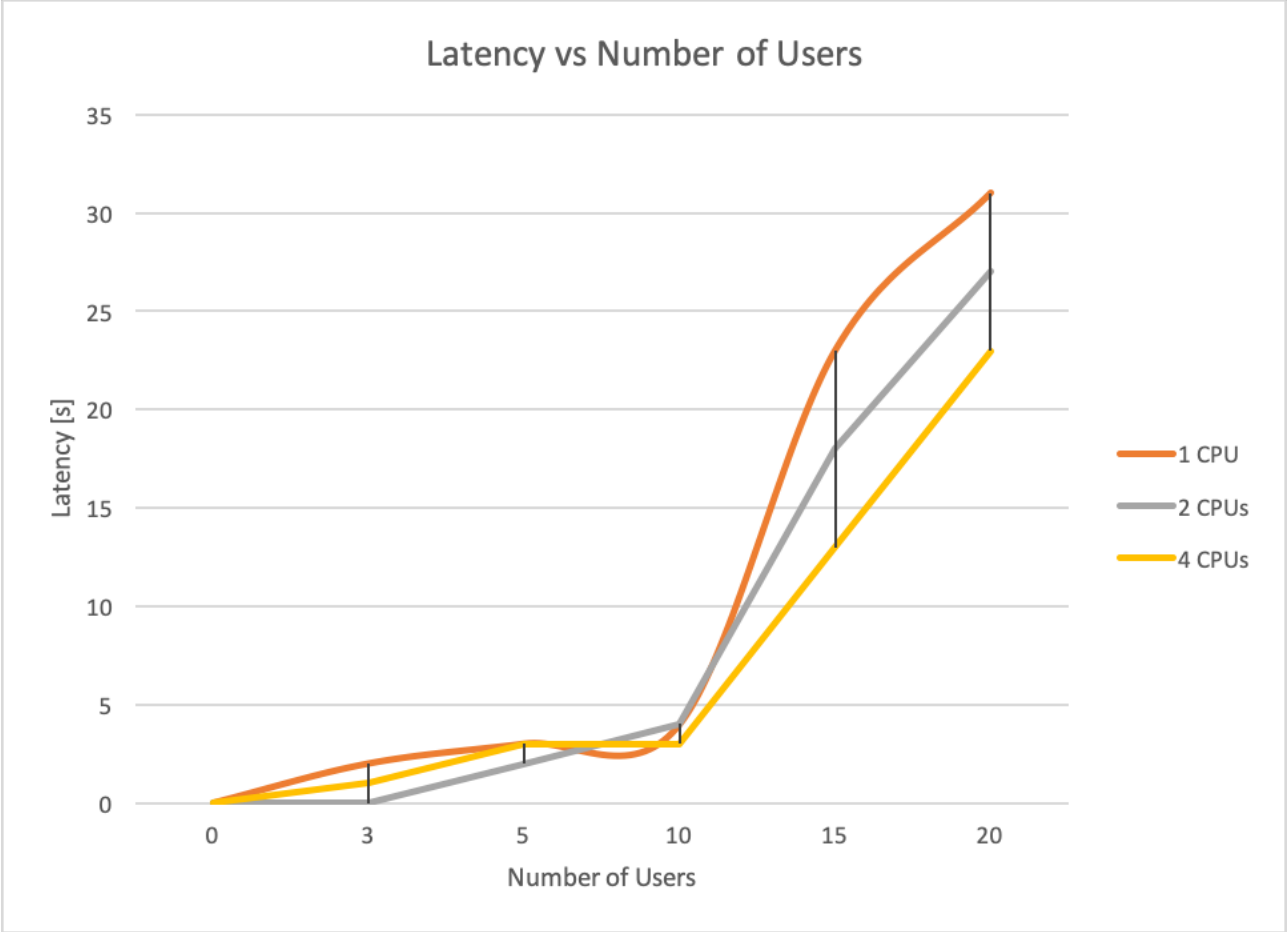
Measuring the performance

Possibilities how to measure performance of a web application can be several, most often it is: Throughput, Latency and CPU usage. All these three types are used to measure performance of our application.

Graphs

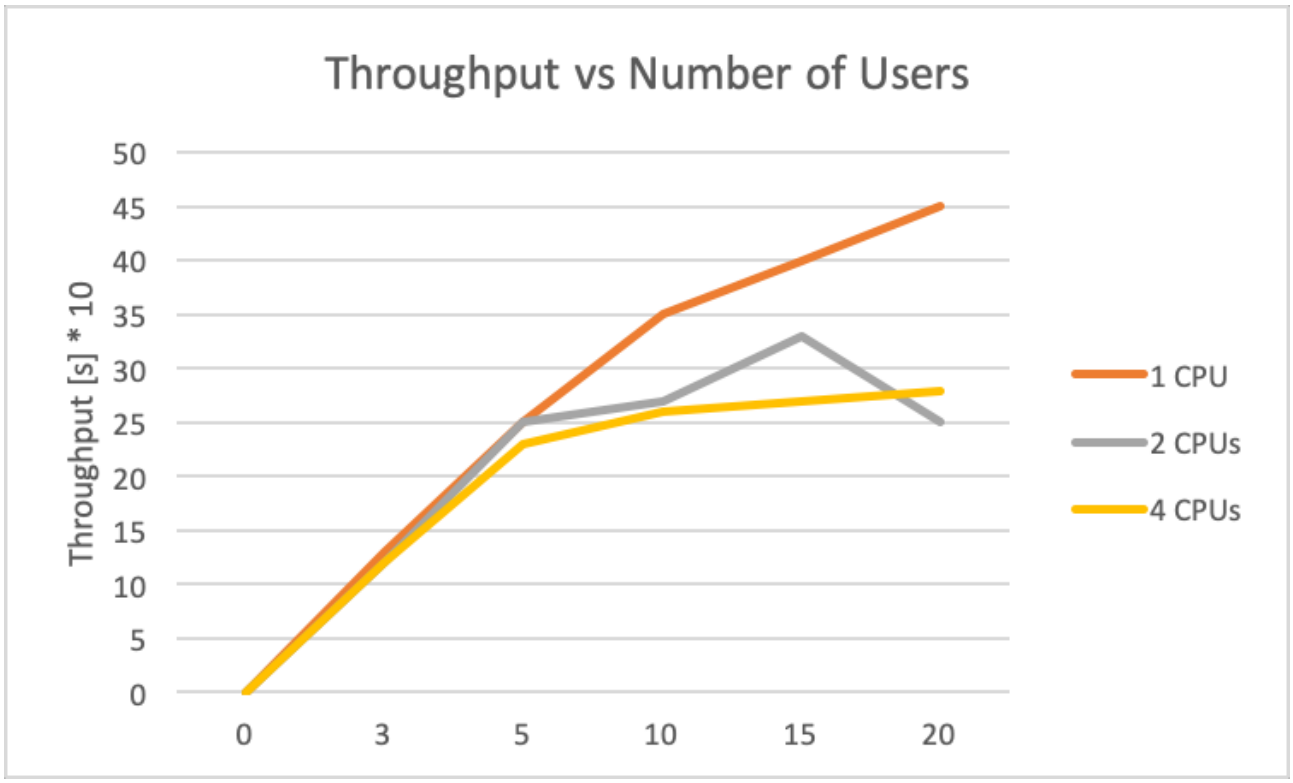
- **Latency x Number of users**

Latency means amount of delay compared to how many users are requesting. The graph shows difference between using just 1, 2 or 4 cores processor.



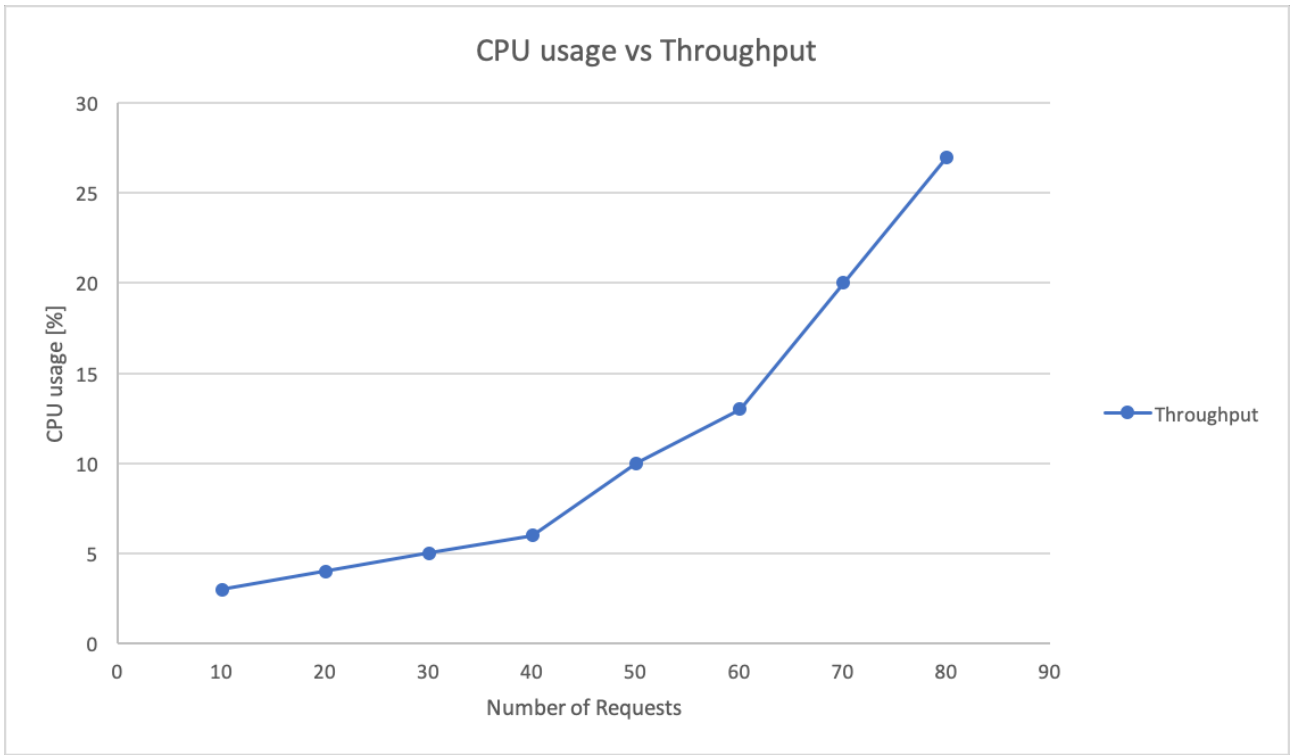
Throughput vs Number of users

Throughput means how much data travel throw successfully. The graph also shows difference between using 1,2 or 4 cores processor as the graph above.



- **CPU usage vs throughput**

This graph shows how the CPU usage grows with increasing number of requests. We got closed to 20%/80 requests which seems to be low, but the usage starts to be increasing more dramatically after more than 150 requests.



- **Memory usage vs throughput**

This graph shows how the memory usage compared to throughput. When the throughput is increasing, almost linearly increases the amount of memory which is allocated.