# Server-Client Exercise

## DOCUMENTATION

The program's scope is to create one Server a Client and make them communicate and exchange multiple messages in a multithread way together. It consists of a Server base class and Client base class.

The server class creates threads in order to service the client. Client base class also creates threads in that case our users. Each user as thread communicates with a Server thread. The program is based on multithreading. Every user sends an array of characters (filled with ".") at the size of about 300 – 2000 KBs. In the beginning client sends a message consisting of

- Hello
- IP & port
- User ID that was provided by the base class

The server then prints the messages as an indication that it received them. After that the server replies back by sending

- Welcome
- Payload (char [])

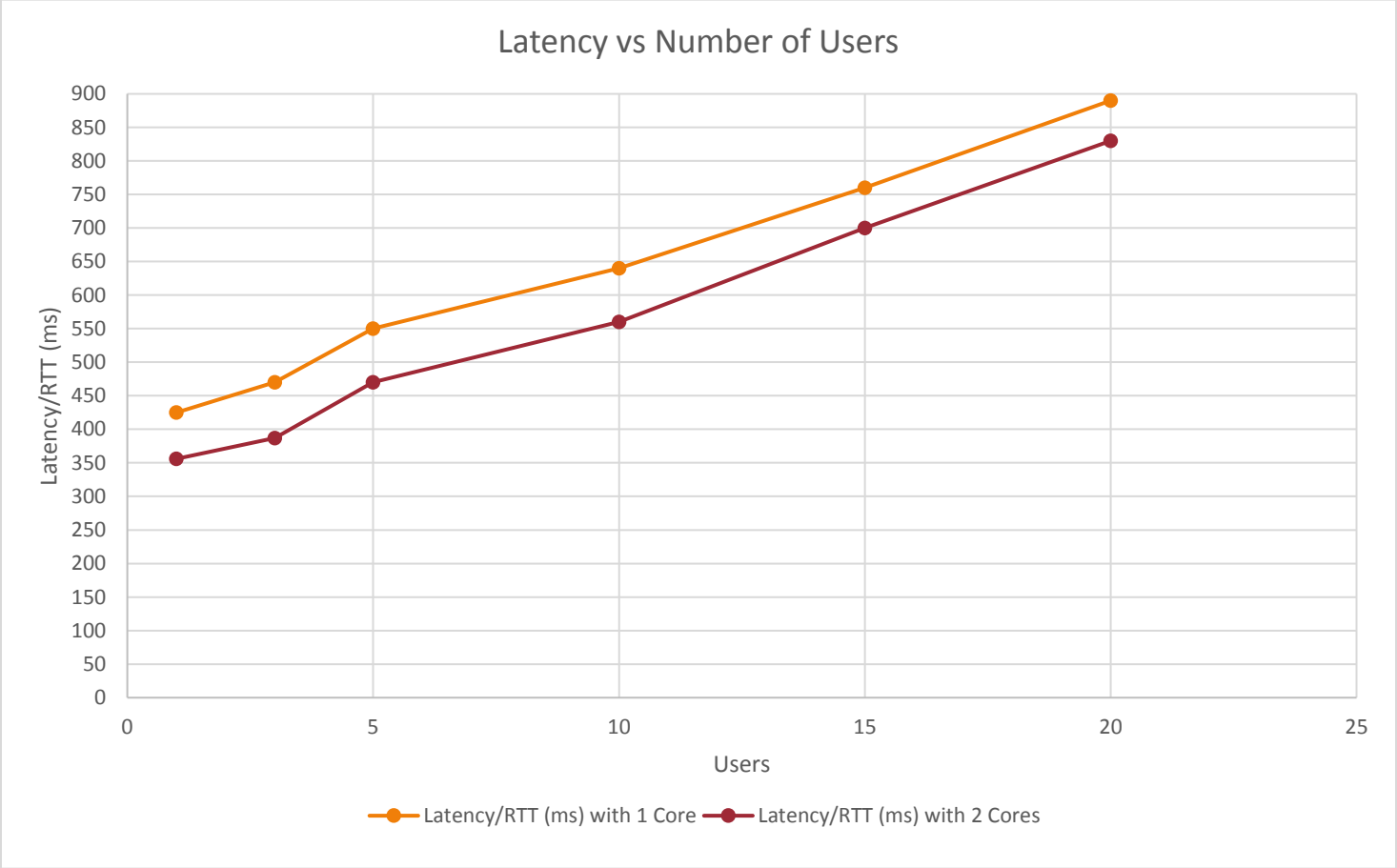
The client then again **proves** that it received the payload by printing a character at the end of the payload which is a dot. Every user sends 300 requests and gets 300 responses back.

Inside our program we use Buffered I/O streams so we can keep the connection going without breaking the sockets. Also all the messages exchanges are **printed**.

## COMMENTS ON PLOTS

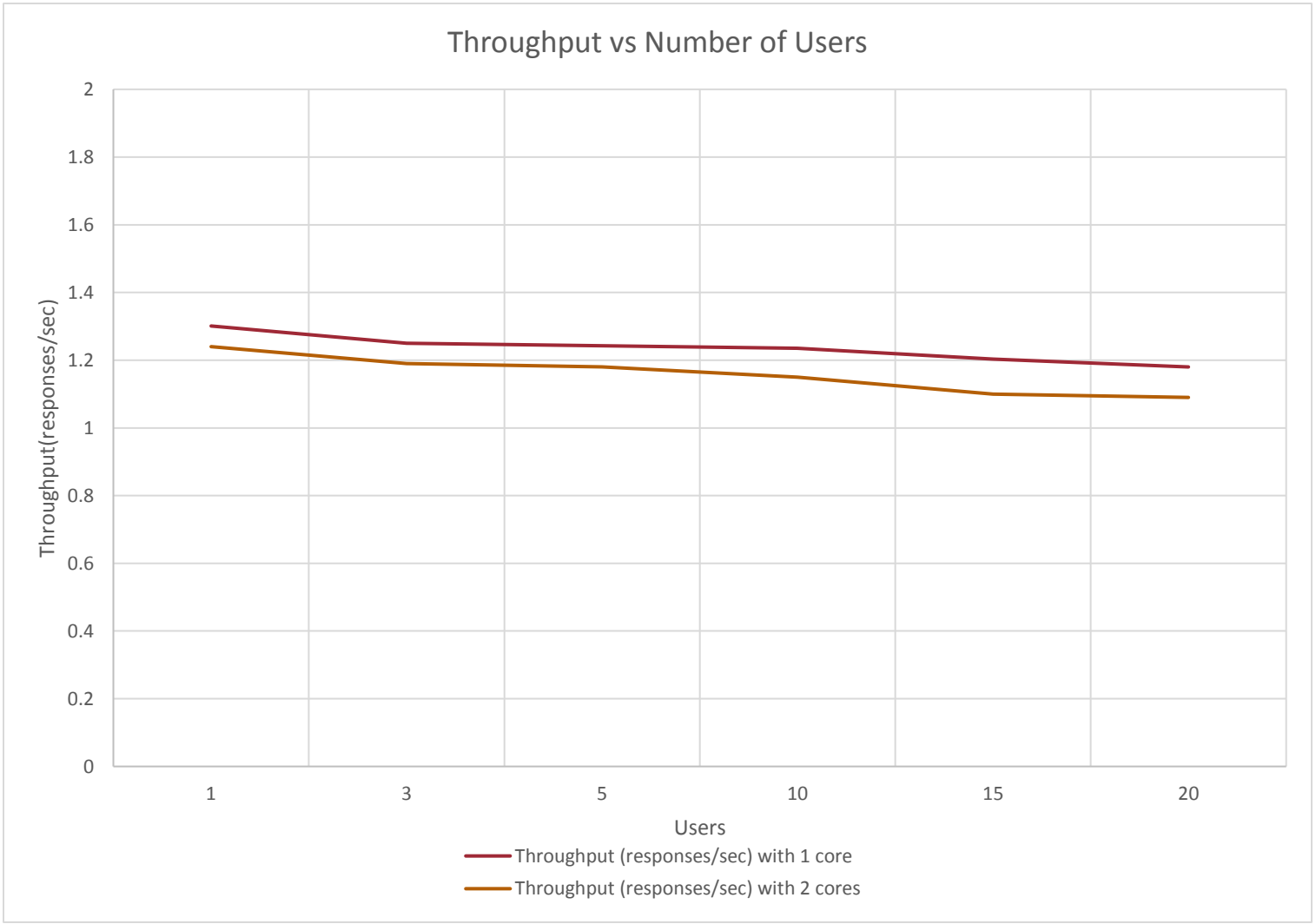
**Plot on latency:**

Latency is obviously increasing as the number of users are increasing. The RTT can start from 300 – 400 milliseconds and can peak up to 900 – 1000.Time for round trip time here seems reasonable. In the begging we had some struggling that it was to slow but now measurements are more convenient. The 2 CPU VM showed slightly better results in reduced latency. The 1 CPU VM was little slower by 100 more or less. It was calculated by measuring the time we need to make one request, send it and get back an answer (round trip time).



**Plot on throughput:**

Throughput is to about 1.3 - 1.1 responses per seconds. The 1 CPU VM has little less throughput in 2 CPU VM. Throughput is slightly decreasing as the users are getting more. We measured time of completed requests divided by time (nanoseconds) that it took to complete each one in an overall duration.



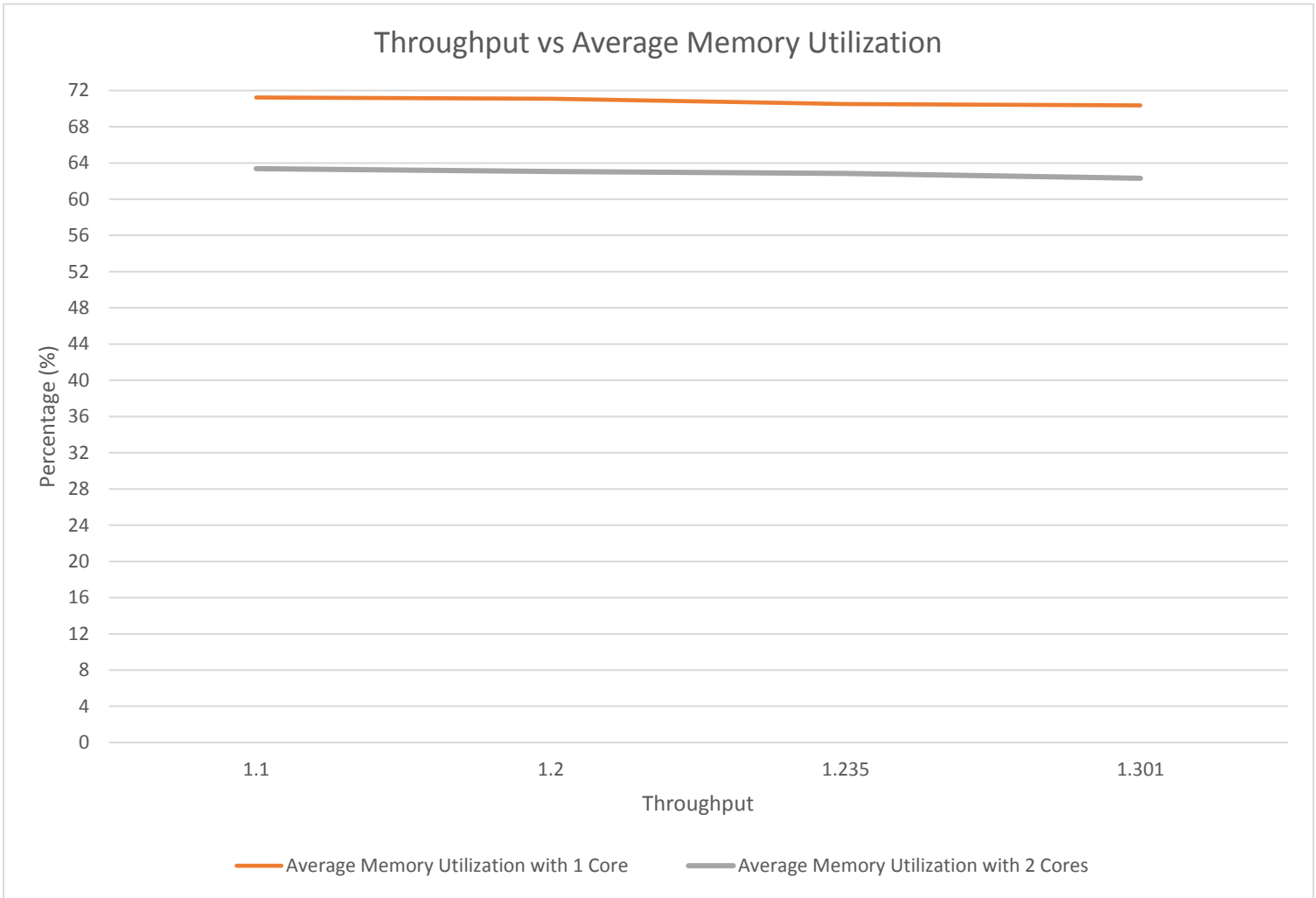
**Plot on Average CPU & Memory Utilization:**

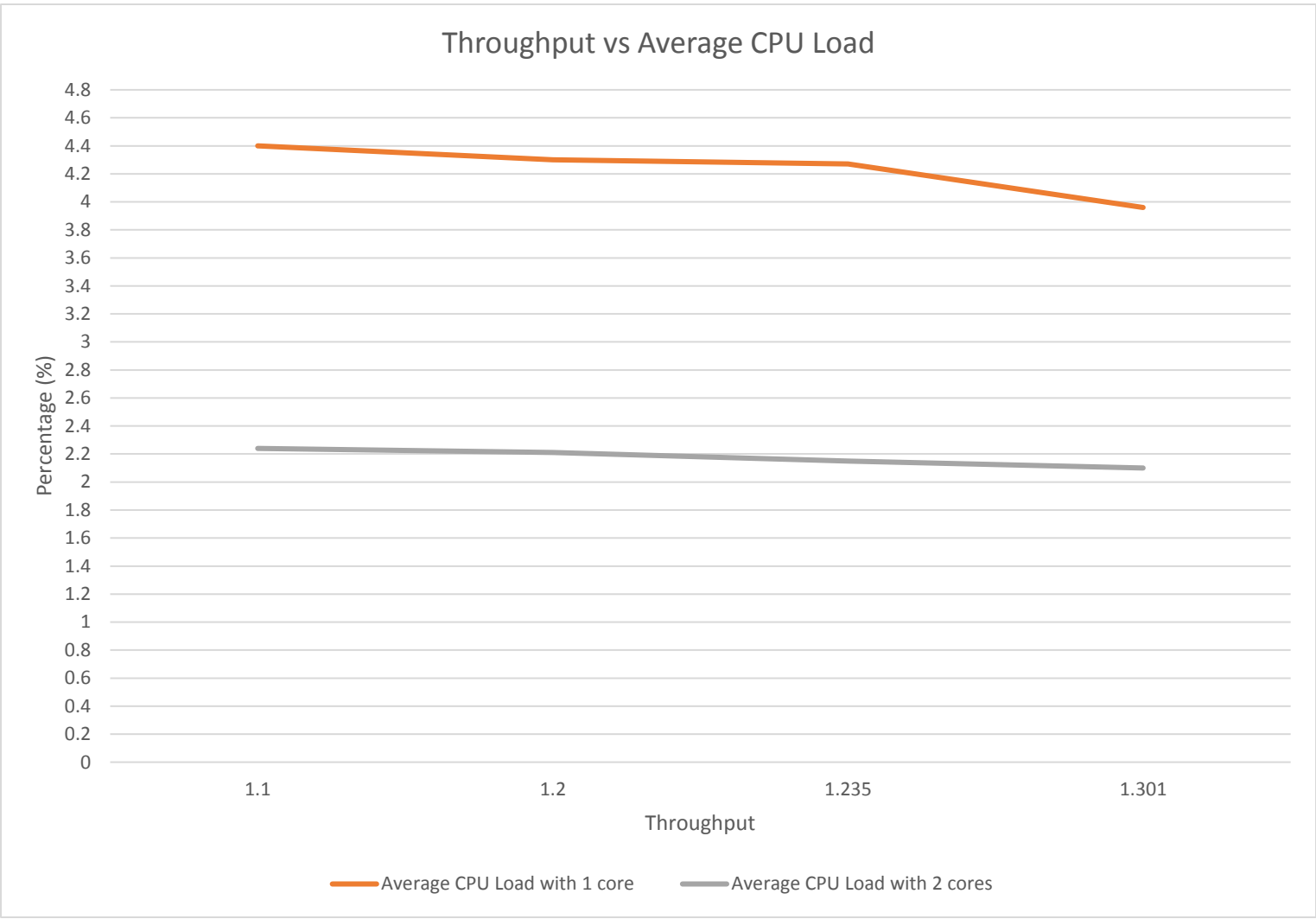
To find the average CPU load we simulated the example with the 20 Users and calculated our means by the help of our statistics. When the back end server program runs we write inside a file average memory utilization and CPU load.

For example in our file the following lines are produced:

No requests: 22 Second/Users: 22 CPU Load: 1.1 Memory Utilization: 87.1

The stats are the average values through a session with an approach of the correspondent throughput at the specific moment.




Throughput vs Average CPU Load
4.8
4.6
4.4
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Percentage (%)
1.1
1.2
1.235
1.301
Throughput
Average CPU Load with 1 core
Average CPU Load with 2 cores