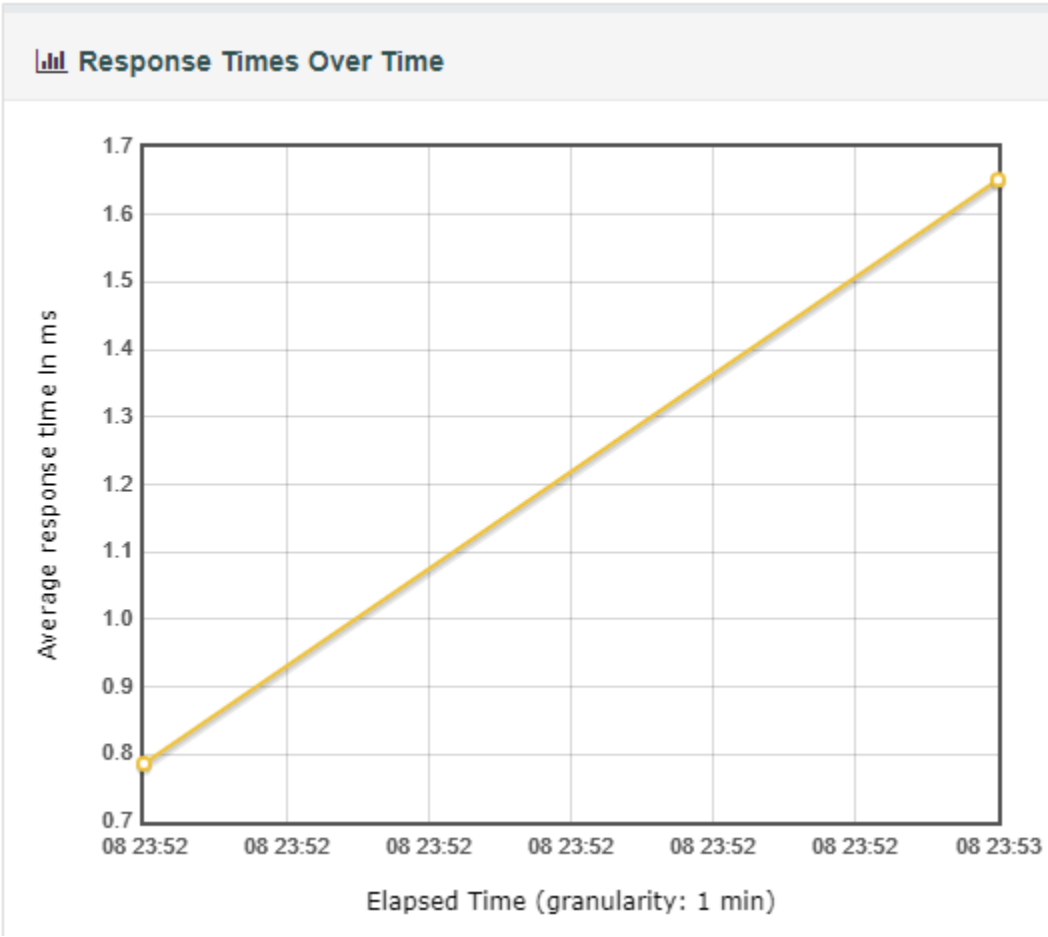
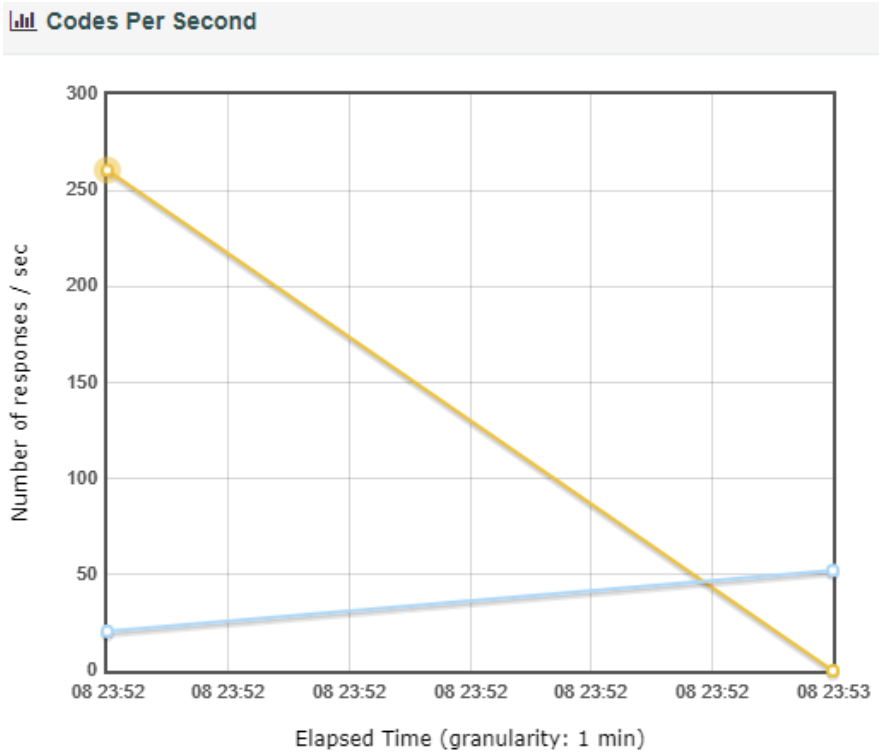
Threaded server

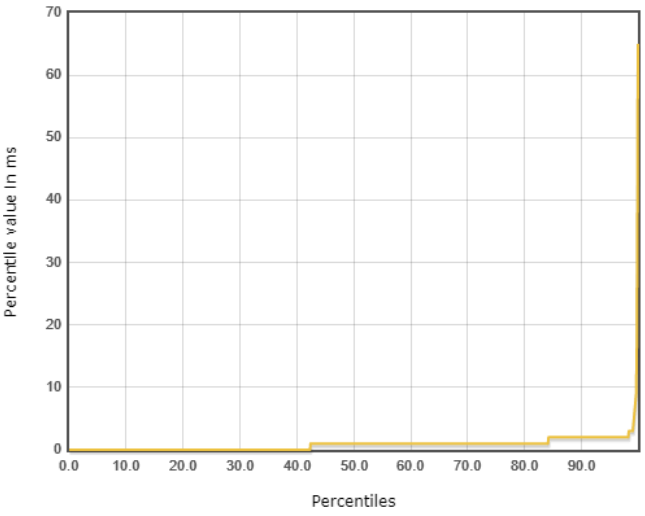
Overall, the server did pretty well. The response times scaled linearly with the number of threads currently being handled:



The primary error was that the socket was currently in use, meaning that the test was receiving a Denial of Service. This is good because the server was still able to service requests after they freed up as well as linearly was able to handle the load:

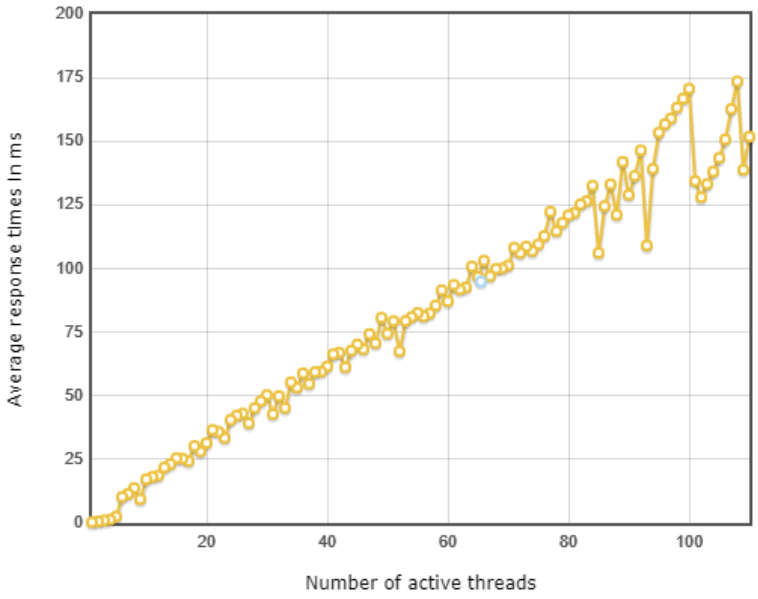


Response time percentiles, however, did exponentially increase at the end (as Windows ran out of ports to use):

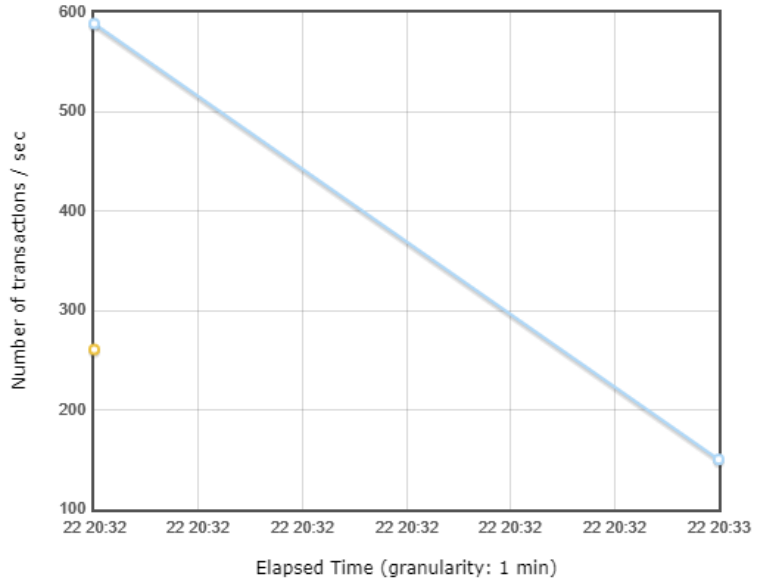


AOI Server

Overall, this server performed better than the threaded TCP server. The average response times vs number of active threads showed a linear growth of response times with the number of active threads. This is a good response because the server didn't show signs of slowing down under a load, except where connections couldn't be established due to operating system errors.



Additionally, throughput scaled down inversely with the number of threads active and number of requests made. This response is good for the same reason as before, that increasing the number of threads doesn't invoke crashing or exponential decrease in response time. This also implies that it's not my server having problems, but rather the operating system allocating sockets:



JMeter lastly reported that all of the errors (100%) were “500/java.net.BindException: Address already in use: connect”, which informed me as to the primary bottleneck of the server. This is further discussed in the lessons learned section.

Errors

Type of error	Number of errors	% in errors
500/java.net.BindException: Address already in use: connect	44347	100.00%

Lessons Learned

As it turns out, most of these DoS errors were coming from Windows itself. This is because Windows normally uses ports 1024-5000 for ephemeral TCP ports and TIMED_WAIT is very long afterwards (with high number of seconds), which means that when we want to have many concurrent users, Windows actually throttles what the server could do because it runs out of ports. This is fixable by modifying the registry on Windows and adding a TCP Max Port key set to 65534.