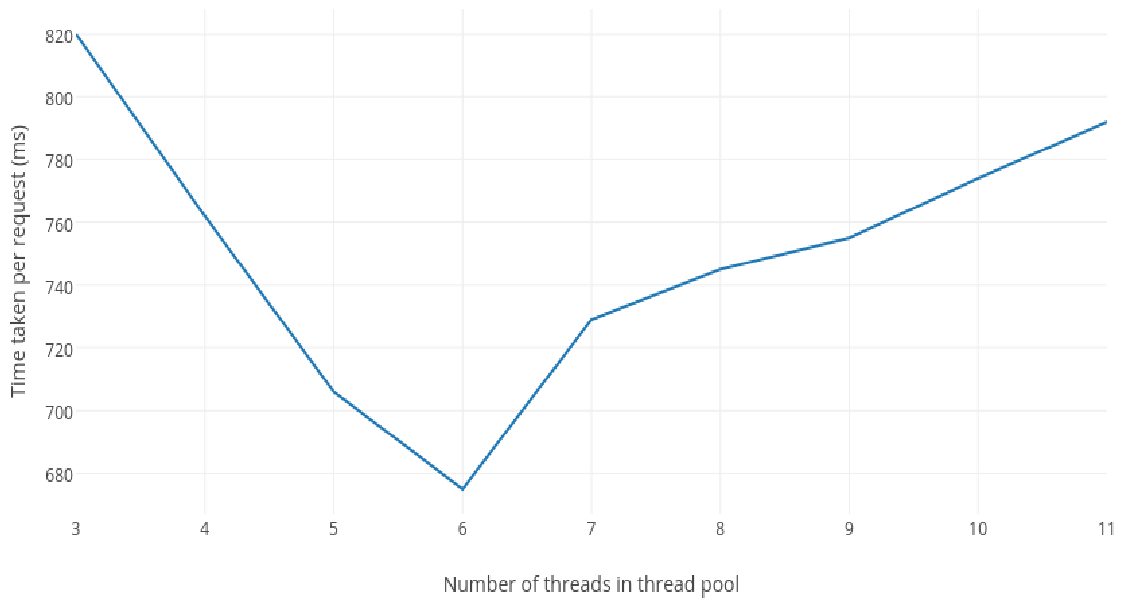
6.4. Performance Testing

I tested the BusyServlet with varying size of the thread pool and found the following relation between time taken per request, Time taken per concurrent request and the number of threads. Note that the tests were carried out by apachebench with **10000 requests and with a concurrency level of 1000.**

Servlet: BusyServlet.java

Number of threads in thread pool	**Time taken per request (ms)**	**Time taken per request (ms)**
3	820	0.821
4	762	0.762
5	706	0.707
6	675	0.675
7	729	0.729
8	745	0.746
9	755	0.756
10	774	0.774
11	792	0.792



As seen in the graph and the table above the time taken per request for the BusySerlvlet is minimum for 6 threads in the thread pool, if number of threads is decreased below 6 the time per request taken increases which is due to the decrease in the level of concurrent processing of requests and when it is increased above 6 then also it takes more time per request due to the limited processing power of the CPU and increased context switching. So the ideal number of thread in the thread pool will be 6 for Busy Servlet. However, for a computationally lesser intensive servlet like demo servlet, the ideal size of thread pool is 10 which I found by a similar analysis. So we can see that a computationally intensive servlet gives the best performance in terms of processing time per request when the thread pool size is lower than the pool size in case of a lighter servlet (like Demo Servlet). This is because for a longer running task each thread takes much more resource in the CPU so more threads (>6 in this case) exhaust the processing power of the CPU. However, for a lighter task like Demo Servlet, single thread consumes less CPU resources so the CPU can accommodate more of these lighter threads, hence ideal thread pool size in this case is greater than 6 (i.e. 10).