555 HW1 M2 Performance Testing EC

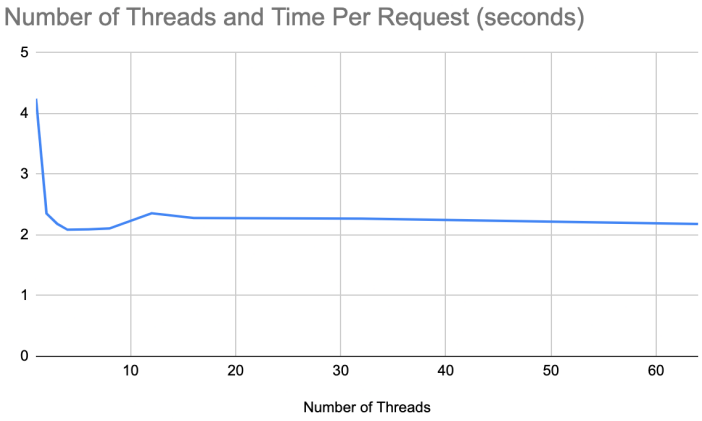
Setup

To stress-test my webserver, I made an endpoint for “GET /factorial/:target” which returns the factorial of an integer passed as the path parameter. The input was 121212, which through testing against a single-threaded instance took ~5 seconds to complete per request.

For my data, I used Apachebench to send 100 concurrent requests at once, and measured how long it took to complete.

Results

Number of Threads	Total Processing Time (seconds)	Time Per Request (seconds)
1	424.491	4.24491
2	235.499	2.35499
3	218.515	2.18515
4	208.598	2.08598
6	209.206	2.09206
8	210.607	2.10607
12	235.655	2.35655
16	227.971	2.27971
32	226.999	2.26999
64	217.969	2.17969



Discussion and Recommendations

We can there's massive improvement ramping up from 1-4 threads, while there aren't many noticeable improvements afterward. This is likely because MacOS will take each thread of a process and put it on a different core of its host machine. Because I'm on a dual-core laptop, where each core has two virtual cores, you would need at least 4 threads to maximize performance.

Looking at Activity Monitor supports this idea. At 1 thread, the average CPU utilization is ~100%. At 2, it's at ~200%. At 3, it's at ~300%, and at 4 it's at ~330%. Why only a 30% bump for the fourth core? That last core is where all the other single-threaded processes are running. My guess is MacOS greedily chooses the least-busy cores to put new threads on.

So, I recommend having at least as many threads as you have cores on your machine. Afterward, you have multiple threads on the same core, and thus they are sharing the same amount of computing resources with some small amount of memory and compute overhead.