The data for the chart was when executing the program at bounds a = 0, b = 1, and n = 1,000,000,000.

Analysis of Speedup vs. n\_threads (p):

Here, we see that adding more threads decreases the execution time of the program, thus increasing the speedup. Multithreading increases our speedup because it divides the workload over n\_threads, allowing for greater CPU utilization. We observe a sublinear speedup, where adding more threads leads to diminishing returns. Some causes for sublinear speedup are thread overhead, such as cache coherency and thread management, and having a sizeable sequential code portion. The highest performance ratio was when we went from using 1 thread to 2 threads.

Analysis of Efficiency vs. n\_threads (p):

Here, we see that the efficiency of adding more threads diminishes. Since the efficiency rate is decreasing, we know that sublinear speedup is happening. If it stayed constant, it would indicate a linear speedup. If it increased, it would indicate a superlinear speedup.