Andrew Huang
amh877
Parallel Computing Spring 2018

Table 1 - $n = 10,000$ Speedup relative to 1 thread

Threads	1	2	5	10	20	40	80	100
	1	4127.82	3439.57	141.139	375.476	3.4274	37.5109	28.7028

Table 2 - $n = 100,000$ Speedup relative to one thread

Threads	1	2	5	10	20	40	80	100
	1	6.694	3.3511	1.9633	1.37912	0.3904	0.18655	0.13976

*Edit - I reran 100k on crunchy3 and I got similar numbers

Explanation for Graph 1:

The greatest speedup happened with 2 threads because I believe the costs of synchronization were less than with more threads, even though the parallelism may have not been the most. That is why I noticed the speedup decreased as the number of threads increased, because even though OpenMP does not have communication costs, there are penalty costs associated with shared memory.

Explanation for Graph 2:

Here, the same patterns as above emerges. Because of the costs of shared memory, the overhead of creating more threads impacts the performance. Again in both cases, I found 2 to be the sweet spot because it both takes advantage of parallelism, while incurring the least amount of synchronization cost among all of the other thread options. For >20 threads, the time became worse than sequential because the overhead costs accumulated.