|  |  |  |
| --- | --- | --- |
| # Orders Processed | Single Thread Time (msec) | Multi Thread Time (msec) |
| 100 | 8542 | 8706 |
| 200 | 15595 | 14838 |
| 300 | 19316 | 18455 |
| 400 | 26279 | 22371 |
| 500 | 32062 | 24773 |

**Data Set:** LargeSet

Looking at my results, it is clear the multithreading is overall the faster implementation under the condition that the dataset is larger. On my particular machine, this was shown starting when I was processing 200 orders. Since creating a new thread creates more overhead, a single-threaded process can complete faster so long as that overhead takes up more time than the benefits of multithreading (concurrent processing) can make up for. As the dataset increases in size, the multithreaded processing completes quicker because despite the overhead costs, the speed of processing is faster than the single-thread sequentially going through the program. Since speed is dependent on both the machine the program is run on and the current state of the machine’s CPUs (other programs running, etc.), the multithreaded program was not extremely faster than the single-threaded processing. On different machines, such as ones with more CPUs, that may increase the multithreaded program speed further.