Stefan Traska

Time to Complete Threaded Processing of Orders

|  |  |  |
| --- | --- | --- |
| Number of Orders | Single Threaded Processing Time (ms) | Multiple Threads Processing Time (ms) |
| 1 | 140 | 151 |
| 5 | 368 | 275 |
| 10 | 599 | 396 |
| 25 | 1054 | 809 |
| 50 | 1946 | 1341 |
| 100 | 3243 | 2890 |
| 150 | 4939 | 3753 |
| 250 | 7837 | 5831 |
| 350 | 10754 | 10527 |
| 500 | 16242 | 14738 |

In order to obtain this data, I used the OrdersProcessor program to calculate the time (in ms) to process the LargeSet orders using both multiple threads and a single thread. The data above shows that using multiple threads proved to process the orders faster than using a single thread. As the number of orders increased, the margin between processing times for multiple threads and a single thread increased, showing that as the number of orders increases, using multiple threads is increasingly effective. We can let s(n) represent the time it takes to complete the processing using a single thread, and m(n) represent the time it takes to complete the processing using multiple threads (with n being the number of orders). Assuming that both s(n) and m(n) follow the pattern in the table above, we would see that . This shows that s(n) is a larger function and has a greater algorithmic complexity than m(n), and proves that using multiple threads is more efficient, as it synchronizes tasks to complete them faster.