Jacob Slattery

Distributed and Cloud Computing

Dr. Ana Stanescu

13 Feb 2019

Threading Efficiency

This paper's goal is to show the effectiveness of multithreading and to show it the problems of using too many threads. Through synchronization the code is able to use multiple threads to work on the single task of finding all perfect numbers within a given range, which in this case is the upper limit of 3 billion. The process works by calculating potential perfect numbers with factors of Mersenne prime, which has been proven to be directly linked to perfect numbers. According to Euclid–Euler theorem, every even perfect number can be derived from $2^{n-1}(2^n - 1)$. Odd perfect numbers have not been proven to not exist, but since none have been found and are not existent within the range of long numbers, the code excludes the possibility of their existence the calculations. Through the method of predicting and then checking potential improvements it is possible for a computer with an i7-7700 CPU to find all perfect numbers up to 3 billion in under 25 seconds even with a single thread. Multithreading is incorporated by using a multiple threads which continue to get the next potential perfect number from a premade list until all numbers have been checked. This assignment took around an hour to code.

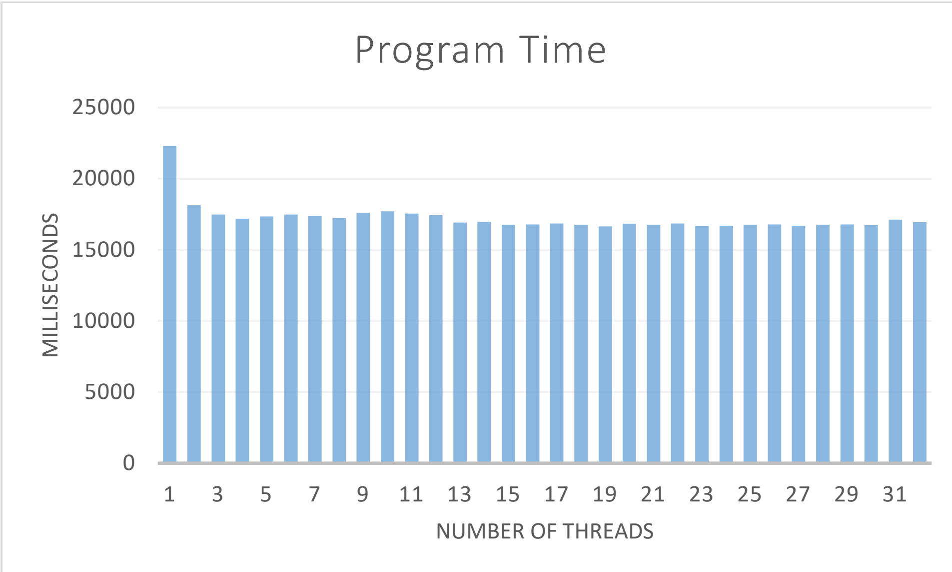
Information Collected

- Running the code on a single thread is noticeably slower than any other test results (See chart "Program Time").
- The jump from one thread to two has the most significantly positive performance impact (See chart "Program Time").
- Adding threads consistently decreases the time needed for the program until it matches the number of physical cores on the PC, after which the efficiency boost per thread is inconsistent but with a slight average trend towards a decrease in time (See chart "Logical Processors").
- In this algorithm, if a core hits 8 threads running on it the trend shifts and increasing the thread count begins to decrease the efficiency on average (See chart "Too Many Threads").
- Increasing the thread count past the number of logical processors will continue to average an increase in efficiency (See chart "Program Time Past Core Count").

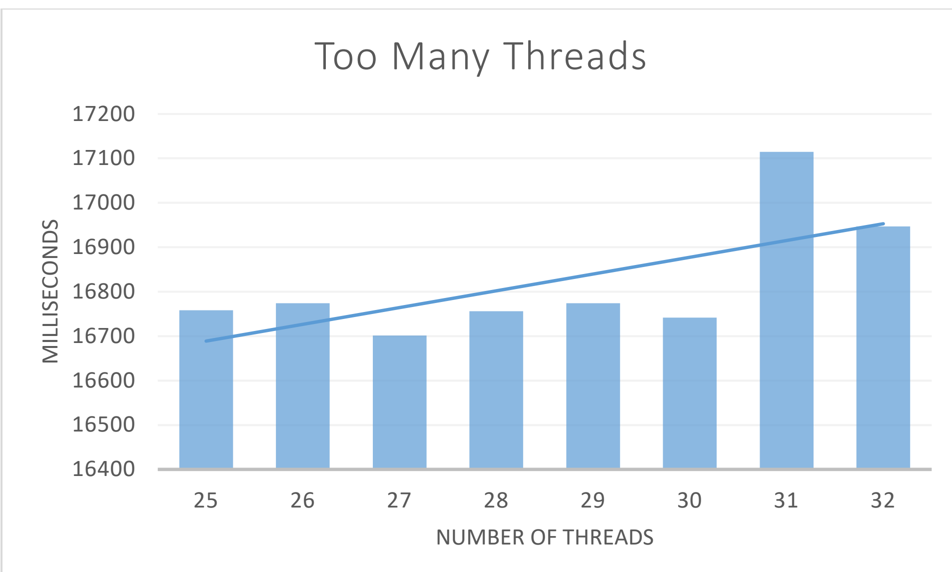
Conclusion

Threading is a very useful tool for increasing the efficiency of a program, but it must be kept in check. In this instance, the performance impact on the program by adding threads past the core count is negligible, and since these threads use up more memory it is reasonable to limit the count of the threads to match the computer's core count in order to get the most out of your computer while saving as much memory as possible.

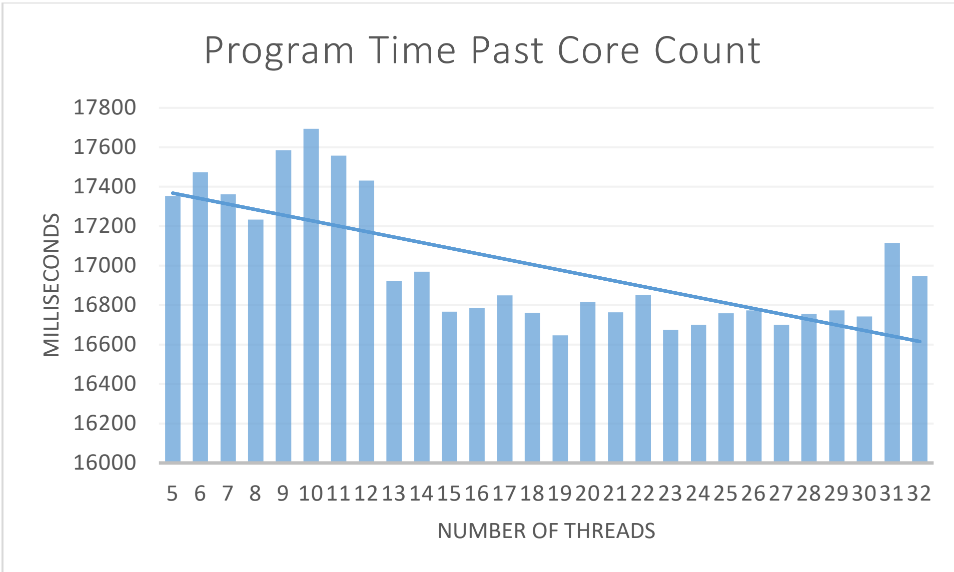
Appendix

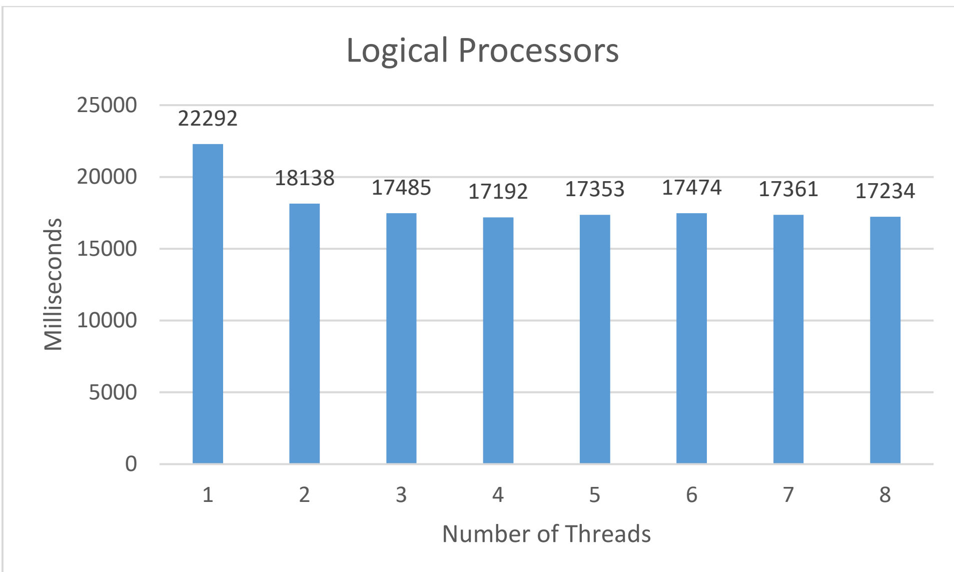


Program Time
25000
20000
15000
10000
5000
0
MILLISECONDS
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
NUMBER OF THREADS



Too Many Threads
17200
17100
17000
16900
16800
16700
16600
16500
16400
MILLISECONDS
25
26
27
28
29
30
31
32
NUMBER OF THREADS





Sources:

"Perfect Number." *Wikipedia*, Wikimedia Foundation, 5 Jan. 2019, en.wikipedia.org/wiki/Perfect_number.

"List of Perfect Numbers." *Wikipedia*, Wikimedia Foundation, 24 Dec. 2018, en.wikipedia.org/wiki/List_of_perfect_numbers.

"Euclid–Euler Theorem." *Wikipedia*, Wikimedia Foundation, 6 Oct. 2018, en.wikipedia.org/wiki/Euclid–Euler_theorem.