CSE-5250, Spring 2023  
Final Assignment: Benchmarking Performance of the Sieve of Eratosthenes

**This assignment will consist of these parts:**

1. **The multi-threaded sieve of Eratosthenes program (30pts)**
2. **MPI benchmarks (80pts)**
3. **Answer these questions (20pts)**
   1. **Where do you think the “sweet spot” is for thread count? Alternatively, at what point does the addition of more threads result in diminishing returns?**
      * The “sweet spot” seems to be around 4 threads, which is the same conclusion I came to while dealing with OpenMP. With 4 processes we can nearly halve the time it takes to process higher workloads while only marginally decreasing efficiency on lower workloads. Diminishing returns are approached fast after 4 threads/processes, such as the 8 threads tested below. While for larger workloads there is still an increase in efficiency it is only a fraction of the increase from 2 to 4 threads.
   2. **Where do you think the “sweet spot” is for workload size? Alternatively, at what maximum value (max value to search for prime numbers) does it become beneficial to run the sieve in parallel?**
      * The “sweet spot” for workload size is around prime numbers up to 100,000 where we see the greatest efficiency increase with increased number of processes. Surprisingly it becomes beneficial to run the sieve in parallel from as primes as few as up to 10,000 while 100,000 is where the truly noticeable change occurs.
4. **Answer this additional question with a few sentences: Between this and the OpenMP-based sieve from the midterm, what can you say are the advantages and disadvantages of both OpenMP and MPI? You may consider the following:**
   1. **Which was easiest to program?**
      * I found that OpenMP was more intuitive, and I was able to code more easily using it. Without needing to specify which part each process is working on it is simpler to write in OpenMP and not have to worry about writing receive and send functions. OpenMP also had a greater impact on performance in my benchmarking.
   2. **Which one requires (or benefits from) specific hardware?**
      * While OpenMP might benefit the most from having more powerful hardware in small workloads such as what we are working with, MPI benefits from its ability to be used in a multi-system environment and make use of the systems in its control to complete its function. While OpenMP can run in an isolated system, MPI requires a network to run furthering this idea that MPI might benefit or require specific hardware.
5. You will be benchmarking the sieve made using MPI. Start with this table, consisting of 12 entries. You may expand the table to include different thread counts or maximum values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MPI Sieve | Primes up to 1,000 (1e3) | Primes up to 10,000 (1e4) | Primes up to 100,000 (1e5) | Primes up to 1,000,000 (1e6) |
| 2 processes | 5277.2μs | 6544.7 μs | 19679.5 μs | 158076.8 μs |
| 4 processes | 5437.8 μs | 5992.8 μs | 12181.2 μs | 74346.3 μs |
| 8 processes | 5835.2 μs | 6453.3 μs | 11336.3 μs | 53953.5 μs |