The Prime Number Algorithm

The algorithm basically finds all prime numbers until number M. It iterates number n two by twos. Firstly, for each n value there is an integer division, which is n/prime[k], and mode calculation, which is n%prime[k]. The algorithm starts with 3 firstly as prime [1] is 3. If n is divisible by a prime number, it guarantees n is not a prime number. Otherwise, it checks n/prime[k] is greater than prime[k]. If it less, the algorithm increments k by 1, takes new prime number and calculates new quotient and remainder values. If it is greater, n is prime, since the algorithm tried all coherent prime numbers. If n is prime, the algorithm adds n to the list, so increments the index of array by one. Otherwise, j is constant in that loop. Finally, an iteration is over, and the algorithm can check the next n value.

Normally the iterations would be until number M. However, the aim is to parallelize the algorithm. Therefore, the functions are parted into two parts. Firstly, we find prime numbers until finding next prime after square root of M. If we try to make all iterations parallel, there will be lack of some prime[k] values.

After calculating sequentially, the functions use their chunk size parameter. There are three functions with distinct schedules. Static, guided, and dynamic methods.

Results

- High M values causes algorithm run longer naturally.
- For low M values number of threads did not affect algorithm much such as 10000.
- For high values, when number of threads are increased, the algorithm runs faster.
- For 1000000 M value, except 100000 chunk size, the algorithm duration with 8 threads has similar values with distinct chunk sizes and scheduling methods.
- For 1000000 M value, 100000 chunk size gives the worst results between different chunk sizes with 8 threads.
- For 1000000 M value, usually guided scheduling method gives the best speed-ups with 8 threads. The only case it is not best definitely is 100000 chunk size. This size needs distributing scheduling least compared to others. Dynamic also gives good speed-ups. For this case the worst is static.

- For 1000000 M value, apparently, 100 chunk size gives the best results with one thread.