University of California, Riverside
Department of Computer Science & Engineering

Title: High-Performance Computing – Project 3

Student Name:
Mahbod Afarin

Student ID:
862186340

Fall 2020

For this assignment we are going to find the prime numbers in range 1 to 10^{10} using sieve of Eratosthenes. The original program is *sieve0.c*, in *sieve1.c*, *sieve2.c*, and *sieve3.c* we are trying to do some modifications and then analyze the results. The *sieve1.c* does not consider the even integers. The *sieve2.c* don't use broadcasting to find sieving prime and every process will use its sieving prime using local computation. The sieving *sieve3.c* will reduce the cache miss using modifying the loops. In the table 1, we can see the run time of *sieve0.c*, *sieve1.c*, *sieve2.c*, and *sieve3.c* for 32, 64, 128, and 256 cores.

For the first modification, we do not consider even numbers because even numbers are not prime. Therefore, if we do not set aside memory space for even number, we can gain higher performance. For the second modification, we eliminate the broadcasting by local computation and each process can compute their own next sieving number. Because the broadcast operation takes a lot of time and in this way, we can have a better performance. For the third modification, we can change the order of the nested loops. In this way, the outer loop iterates more than the inner loop and the programs becomes cache friendly because the loop which iterate more, is now in the outer. In this way we can save a significant time compared to the previous version.

Table 1: The execution time for different algorithms and different cores

Algorithm	32 cores	64 cores	128 cores	256 cores
Sieve0	27.895410	26.271435	7.093644	6.482263
Sieve1	13.880482	12.927551	3.561341	3.155362
Sieve2	13.607893	12.983536	3.459118	3.121277
Sieve3	6.377443	3.326938	1.604294	0.797720

As we can see in the table 1, seive3 has the best performance compare to the other algorithms and it is because of the improving the cache hit rate and minimizing the broadcast for seive3. Seive0 has the worst performance because it does not have any kinds of optimization. Also, as it is clear from the table, increasing the number of cores will decrease the execution time spectacularly and it means that the algorithms can run in parallel in different cores and we can parallelize them, but the execution time is not reduced by half with when we double the number of the cores. From 32 cores to 64 cores and 128 cores to 256 cores, it will reduce the execution time by less than double, but for 64 to 128, it will reduce the execution time by more than double and the reason is that for 128 cores the program can be parallelize better and it depends on the algorithm, inputs, and the parallelization overhead. In addition, each optimization will decrease the execution time spectacularly. In figure 1, we have

shown the visual view of the results. The total number of prime numbers is 455052511.

In the figure 1, we can see the algorithms and their execution time in one chart. As we can see in the figure 1, the execution time of seive0 is biggest and the execution time of sieve3 is smallest. The blue bar is for 32 cores, the orange bar is for 64 cores, the gray bar is for 128 cores, and the yellow bar is for 256 cores. As the number of cores increases, the execution time will decrease.

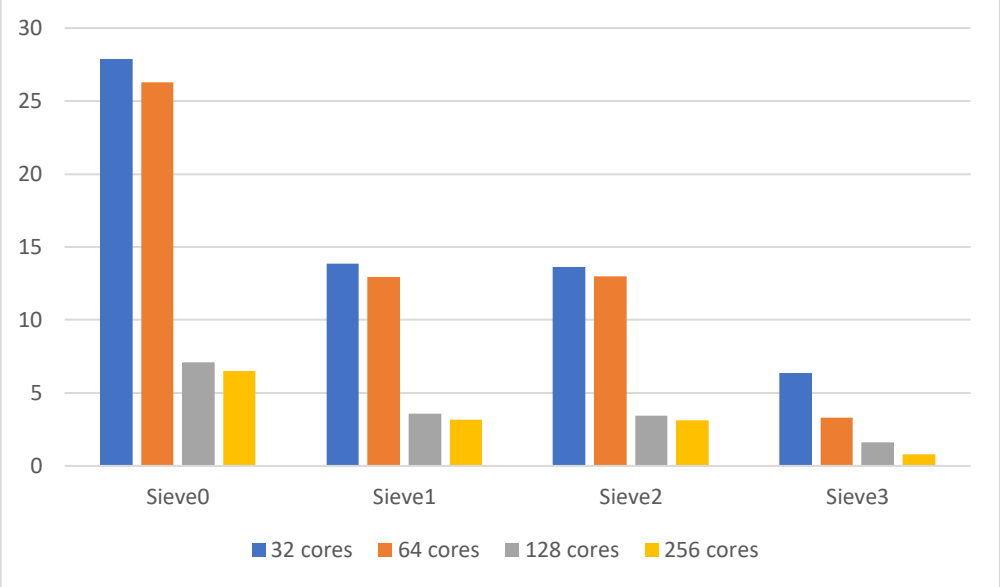

Figure 1: Comparison of execution time between different algorithms

In figure 2, we can see the result of the execution in terminal.

```
[mafar001@tardis result]$ cat final.txt

sieve0:
The total number of prime: 455052511, total time:  27.895410, total node 32
The total number of prime: 455052511, total time:  26.271435, total node 64
The total number of prime: 455052511, total time:   7.093644, total node 128
The total number of prime: 455052511, total time:   6.482263, total node 256
sieve1:
The total number of prime: 455052511, total time:  13.880482, total node 32
The total number of prime: 455052511, total time:  12.927551, total node 64
The total number of prime: 455052511, total time:   3.561341, total node 128
The total number of prime: 455052511, total time:   3.155362, total node 256
sieve2:
The total number of prime: 455052511, total time:  13.607893, total node 32
The total number of prime: 455052511, total time:  12.983536, total node 64
The total number of prime: 455052511, total time:   3.459118, total node 128
The total number of prime: 455052511, total time:   3.121277, total node 256
sieve3:
The total number of prime: 455052511, total time:   6.377443, total node 32
The total number of prime: 455052511, total time:   3.326938, total node 64
The total number of prime: 455052511, total time:   1.604294, total node 128
The total number of prime: 455052511, total time:   0.797720, total node 256
[mafar001@tardis result]$
```

Figure 2: Final results which shows the execution time and number of nodes