Parallel Prime Number Generation.

CSE 633 – PARALLEL ALGORITHMS (SPRING 2020)
PRESENTER: PRIYESH PATEL
INSTRUCTOR: DR. RUSS MILLER, DR. M. D. JONES
Outline

• Why generate Prime numbers?
• Sieve of Eratosthenes algorithm
• Parallel approach
• Data distribution
• Results
• Observations
• Challenges
Why generate prime numbers?

• Several cryptography algorithms like RSA or Diffie Hellman key exchange are based on large prime numbers.

• Computationally large prime number is likely to be a cryptographically strong prime.

• However, as the length of the cryptographic key values increases, this will result in the increased demand of computer processing power to create a new cryptographic key pair.

• Prime numbers also hold an important place in Quantum Mechanics.
Sieve of Eratosthenes algorithm

Source: https://www.realclearscience.com/
Sieve of Eratosthenes algorithm

```c
void sieve(int N) {
    bool isPrime[N+1];
    // initialize the boolean array to true
    for(int i = 0; i <= N; ++i) {
        isPrime[i] = true;
    }
    isPrime[0] = false;
    isPrime[1] = false;
    // If N is prime, its factor will not be greater than sqrt(n)
    for(int i = 2; i * i <= N; ++i) {
        if(isPrime[i] == true) {
            // Mark all the multiples of i as composite numbers
            for(int j = i * i; j <= N; j += i)
                isPrime[j] = false;
        }
    }
}
```
Parallel approach (data distribution)

- Block data decomposition
- If \( n \) is the number of elements and \( p \) be the number of processing elements,
  - Divide data(range) into \( n/p \) contiguous blocks of equal size.
  - Let \( i \) denote the rank of the process
    - Range start = \( i \times (n/p) \)
    - Range end = \( (i + 1) \times (n/p) - 1 \)
Parallel approach (Algorithm)

1. low = rank*(N/size)
2. high = (rank + 1)*(N/size) - 1
3. Initialize/mark $\text{isPrime}$ array in the range low to high with true
4. Find the next marked global minimum value.
5. If next > sqrt(N), GOTO step (8)
6. Broadcast minimum next value to all processors.
7. Unmark all the multiples of next in the range low to high.
8. Apply reduce operation on each of the processor's array
Parallel approach (data consolidation)

- Use reduction operation on every processor’s Boolean array.
- Use MPI_REDUCE(sendbuffer, recvbuffer, count, datatype, op, root, comm)
  - Count = number of total elements
  - Op = MPI_BAND (bitwise AND operation)
  - Root = Processor 0 will receive final array when sieve’s algorithm ends in the recvbuffer.
Results

Sequential Algorithm (1 PE)

<table>
<thead>
<tr>
<th>Input data</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.00002</td>
</tr>
<tr>
<td>1,000</td>
<td>0.00018</td>
</tr>
<tr>
<td>10,000</td>
<td>0.00161</td>
</tr>
<tr>
<td>100,000</td>
<td>0.0195</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.2733</td>
</tr>
<tr>
<td>10,000,000</td>
<td>0.41375</td>
</tr>
<tr>
<td>100,000,000</td>
<td>6.570805</td>
</tr>
</tbody>
</table>
Data: 10,000,000

<table>
<thead>
<tr>
<th>Size: 2</th>
<th>Size: 4</th>
<th>Size: 8</th>
<th>Size: 10</th>
<th>Size: 16</th>
<th>Size: 32</th>
<th>Size: 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.03736</td>
<td>0.02008</td>
<td>0.00725</td>
<td>0.00720</td>
<td>0.05012</td>
<td>0.05552</td>
</tr>
</tbody>
</table>
Observations

• Smaller number of nodes couldn’t handle more than 1 billion data.
• The run time was 3.2128 for 10 billion data when run on 128 node.
• As the input number gets bigger, the number of prime numbers found decreases and hence the communication between the processors also decreases.
• Most of the numbers are already cancelled in each of the processor ranges.
• The algorithm does scale well in parallel, if the input data increases.
Challenges

• Storing values that exceed the double datatype range.
• Getting the Boolean array space allocated for large numbers.
• Finding the first number in the processor’s range that is the multiple of received prime number.
Thank You For Listening. Questions?

Fun fact:
The largest known prime number is $2^{82,589,933} - 1$, a number which has 24,862,048 digits.