Analyzing the Sieve of Atkin: A Modern Approach to Prime Number Generation

Olta Bytyci
Ruslan Kysil
Baran Sheykholeslami

January 17, 2024

Abstract

This research paper presents an exploration of the Sieve of Atkin algorithm. We analyze the algorithm's performance and efficiency for generating prime numbers using parallel computations up to 10^{10}.

1 Introduction

Prime numbers, integral in mathematics and computer science, are especially crucial in cryptography and algorithmic analysis. Traditional methods like the Sieve of Eratosthenes, while historically significant, face challenges of efficiency and scalability. The Sieve of Atkin, developed in the early 21st century, offers a modern, efficient alternative. This paper aims to thoroughly analyze the Sieve of Atkin, exploring its mathematical underpinnings and practical application in a C++ implementation.

2 Sieve of Atkin

2.1 Algorithm Overview

Provide a detailed explanation of the Sieve of Atkin, including its mathematical basis involving quadratic forms and modular arithmetic. Explain how it identifies potential prime candidates and its process for filtering out non-primes using concurrency.

2.2 Algorithm Complexity

The Sieve of Atkin is an optimized sieve algorithm for finding prime numbers and has a theoretical time complexity of $O(N/\log\log N)$.

2.3 Introducing parallelism

We introduced parallelism via a thread pool to distribute computational work across multiple threads, reducing the total runtime by using multi-core CPU architectures. Each loop in the algorithm is divided into smaller tasks for concurrent execution by multiple threads, potentially speeding up the computation.

The Sieve of Atkin uses specific mathematical conditions to target numbers with a higher likelihood of being prime, reducing the computational workload. Post marking potential primes, the algorithm eliminates multiples of squares, further reducing the number of candidates.

2.4 1st Condition: $n = 4x^2 + y^2$

- **What It Finds:** Selects numbers of the form $4x^2 + y^2$ with an odd number of solutions where $n\%12 = 1$ or $n\%12 = 5$.

- **For Loop Condition:** Runs for x starting from 1 until $4x^2$ is less than or equal to the limit.
- **When and Why It Stops:** Stops when $4x^2$ exceeds the limit to focus only on numbers within the desired range.
- **What It Skips:** Excludes numbers not fitting $4x^2 + y^2$ or the modulo conditions.
- **Efficiency:** Efficient at finding prime candidates missed by simpler sieves.

2.5 2nd Condition: $n = 3x^2 + y^2$

- **What It Finds:** Looks for numbers of the form $3x^2 + y^2$ satisfying $n\%12 = 7$ with an odd number of solutions.
- **For Loop Condition:** Runs for x starting from 1 until $3x^2$ is within the limit.
- **When and Why It Stops:** Stops when $3x^2$ is greater than the limit.
- **What It Skips:** Skips numbers not conforming to $3x^2 + y^2$ or the modulo condition.
- **Efficiency:** Further refines the set of potential primes efficiently.

2.6 3rd Condition: $n = 3x^2 - y^2$, $x > y$

- **What It Finds:** Targets numbers of the form $3x^2 - y^2$ with $x > y$ and $n\%12 = 11$.
- **For Loop Condition:** Runs for y starting from 1, checking x values from $y+1$ until the formula is within the limit.
- **When and Why It Stops:** Stops when $3x^2 - y^2$ exceeds the limit.
- **What It Skips:** Excludes numbers that do not match the formula or the modulo condition.
- **Efficiency:** Efficient in identifying primes in a specific numeric pattern.

2.7 General Remarks

These conditions efficiently mark potential prime numbers, with the algorithm later eliminating non-prime numbers. The combined use of these conditions allows for a more efficient generation of primes compared to traditional methods, particularly on a large scale with the advantage of parallel processing.

2.8 Advantages Over Traditional Methods

The Sieve of Atkin offers improved computational efficiency and memory usage over traditional methods like the Sieve of Eratosthenes. Its ability to effectively leverage modern multi-core processors for parallel execution enhances its performance, particularly for large datasets.

3 Implementation and Analysis

3.1 C++ Implementation

The C++ implementation of the Sieve of Atkin exemplifies the algorithm's practical application, showcasing the utilization of advanced programming concepts and data structures to enhance both efficiency and scalability. Using a thread pool for parallel task execution across multiple CPU cores adds another layer of sophistication to the implementation. Upon closer inspection of the basic algorithm, it becomes apparent that the three rules can be applied independently. They essentially toggle the Boolean values in cells. To ensure interference freedom in this parallel execution, we incorporate atomic bool types, guaranteeing singular access to memory without incurring additional memory overhead. This thoughtful approach enhances the overall performance .

3.2 Performance Analysis

We generated primes up to 10^{10} on a machine described bellow:

- OS: Ubuntu 22.04 LTS
- RAM: 32gb drr4 2666mhz
- CPU: Intel core i5-9400f
- Time: Approximately 2 hours

4 Conclusion

Summarize the key findings of the paper, emphasizing the Sieve of Atkin's efficiency and its significance in modern prime number generation. Reflect on the implications of these findings for the field of computational mathematics.

References

C++ documentation
Comparison between the Sieves