CS3510 - Operating System

Assignment 1

Soham Rajesh Pawar
CS22BTECH11055

November 27, 2023

1 Introduction

In the realm of operating systems, the efficient utilization of resources is paramount. This report delves into the implementation of a program in C that explores the world of tetrahedral numbers, leveraging parallel processing to enhance performance. The assignment involves breaking down the task among multiple child processes to compute tetrahedral numbers within a given range.

2 Coding Approach

The program, implemented in C, takes a novel approach to find tetrahedral numbers efficiently. Utilizing shared memory for communication between processes enhances parallelism, allowing for a more streamlined computation process. The assignment was approached systematically, dividing the workload among multiple child processes to exploit the capabilities of modern multi-core systems.

2.1 Steps

1. Read `N` and `K` from the input file to define the computation range and the number of child processes.
2. Allocate shared memory segments to facilitate communication between the main process and child processes.
3. Create `K` child processes, each assigned a unique range of numbers to check for tetrahedrality.
4. Distribute the workload based on the process ID, ensuring each process examines a distinct subset of numbers.
5. For each number in its allotted set, a process records its tetrahedrality status in its respective log file. Additionally, on identifying a tetrahedral number, the process stores the number in its shared memory.
6. The main process waits for the completion of all child processes.
7. Access the shared memory of each process, recording the identified tetrahedral numbers (if any) in the main log file.

3 Verification

The verification process involves rigorous checks to validate the correctness of the program's output. Special attention is given to handling edge cases and potential pitfalls that might affect the accuracy of the results.

4 Conclusion

In conclusion, the implemented program showcases the efficiency of parallel processing in tackling computational tasks. The analysis of execution times and the impact of varying parameters provides valuable insights into the scalability and performance of the solution.

5 Graphs

5.1 Maintaining K at 8
Time vs Size(2^{3N}) :

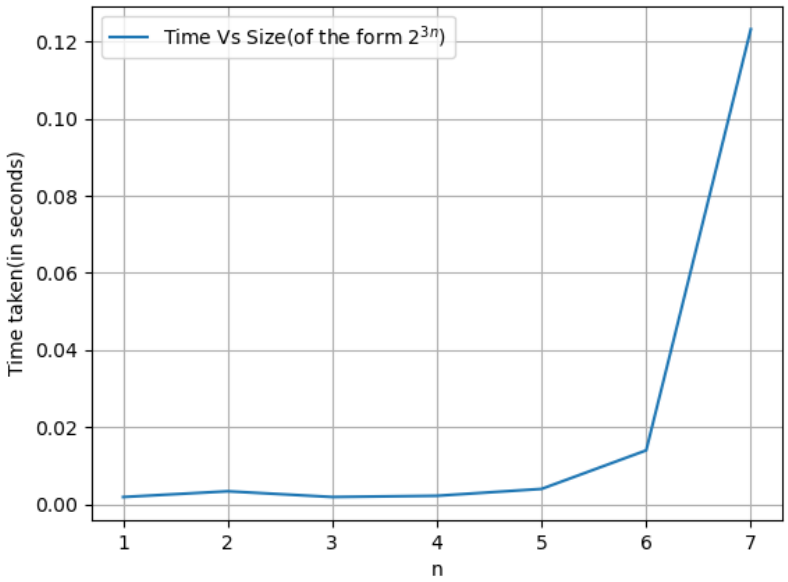

Figure 1: Relationship between execution time and input size with K set to 8.

5.2 Maintaining Size at 10,00,000
Time vs Number of Processes (K) :

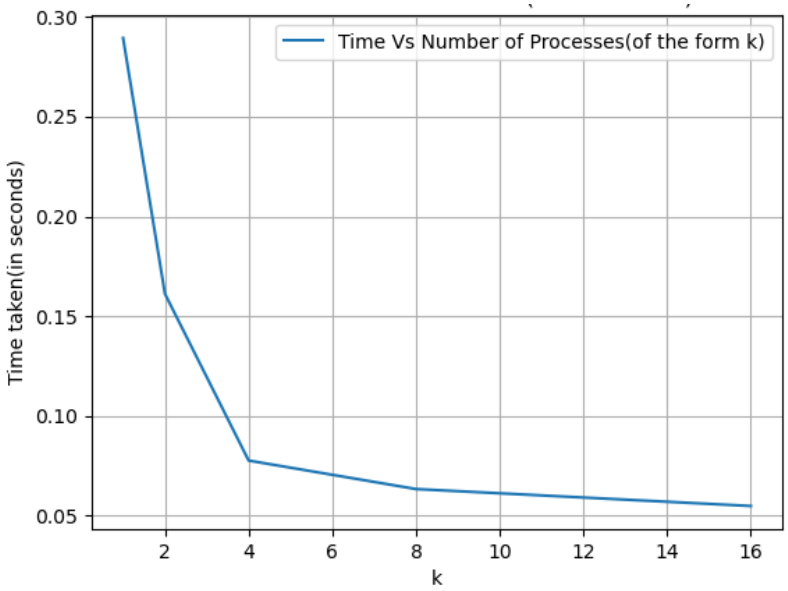

Figure 2: Impact of varying the number of processes (K) on execution time with a fixed input size.