CS 531 HW 2

Hritvik

November 8, 2025

1 Introduction

Prefix sum is a common algorithm used for sorting, histogram generation, and data compaction. The goal of this experiment is to analyze the performance of three implementations:

- Serial prefix sum ($O(N)$)
- Parallel prefix sum using the $O(N \log N)$ algorithm
- Parallel prefix sum using an optimized $O(N)$ approach (Balanced Binary Tree)

2 Experimental Setup

All experiments were performed using an input size of N = 33,554,432 elements. Each implementation was tested with 1, 4, 16, 32, 64, and 128 threads. Execution times were measured in seconds.

3 Results

Table 1 summarizes the execution times for each implementation.

Table 1: Execution time (seconds) for prefix sum implementations

CPUs	Serial $O(N)$	Parallel $O(N \log N)$	Parallel $O(N)$
1	0.0292	0.3528	0.1693
4	0.0302	0.1169	0.0610
16	0.0312	0.0804	0.0440
32	0.0306	0.0352	0.0168
64	0.0297	0.0314	0.0109
128	0.0242	0.0276	0.0081

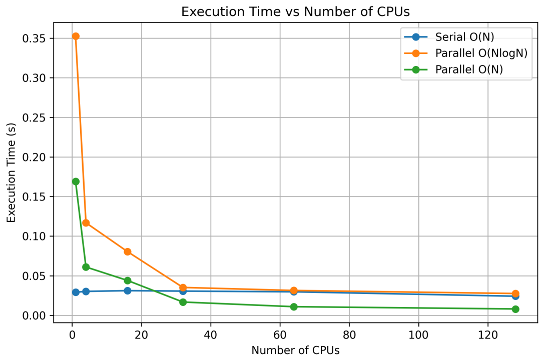

Figure 1: Execution Time vs Number of CPUs

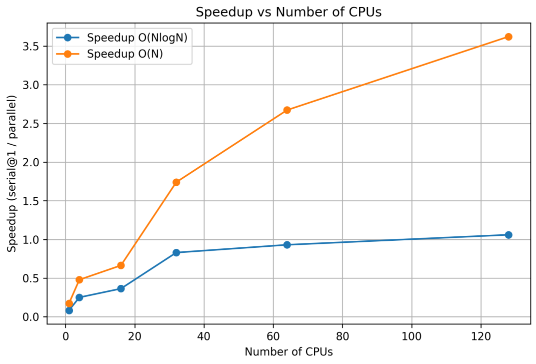

Figure 2: Speedup vs Number of CPUs (Baseline = Serial time)

4 Analysis

As shown in Figure 1, the parallel implementations significantly reduce execution time as the number of CPUs increases. The $O(N)$ parallel version achieves the fastest performance due to its better work efficiency compared to the $O(N \log N)$ method.

Figure 2 illustrates near-linear speedup up to 32 CPUs, with diminishing returns beyond that point.

Overall, the $O(N)$ parallel prefix sum shows excellent scalability and efficiency compared to both the serial and $O(N \log N)$ implementations.

5 Conclusion

This experiment demonstrates the advantages of parallelization in prefix sum computation. The optimized $O(N)$ implementation consistently outperformed the $O(N \log N)$ version, achieving more than a 3x speedup at 128 CPUs.