**Runtime Analysis**

1. **Single Thread CPU**

The time complexity is O(n2) because the loops requires that each index sums all prior indices.

1. **Naïve parallel**

Each thread computes the sum for its portion of the array. In the naive approach, a thread may need to perform a loop from the start of its segment to the end of the segment, which could lead to reading previously computed values from other threads. If the total number of elements is n and each thread computes the prefix sum independently, the naive parallel scan would typically require O(n) work per thread. The total amount of work done across all threads can still be considered O(n), as they are working in parallel

1. **Recursive Doubling Implementation**

At each step, the distance between elements doubles, so the number of steps required to cover the entire array is log2n. In each step, all threads can perform their computations in parallel, so the work done per step (in terms of communication or memory operations) is constant, which is O(log n)