CPU Algorithm Design

Exercise 2 **Students:** Vishal Mangukiya, Konstantin Benz

2.1 Performance Analysis of Reduce and Transform

The benchmark results in Figure 1 and Table 1 show a clear performance improvement when using SIMD-based implementations over the baseline STL version. The reduce benchmarks reveal that SIMD vectorization significantly increases throughput, particularly in custom-controlled vertical implementations.

The baseline STL version (`benchReduceStl`) achieves only about 9.93 GB/s, which serves as a reference point for comparison. In contrast, SIMD-based and OpenMP-assisted variants such as `benchReduceSimdStl`, `benchReduceOmpIterator`, and `benchReduceOmpRange` all reach approximately 16 GB/s, demonstrating the benefit of vectorized addition and loop-level optimizations. The highest performance is observed in `benchReduceSimdXVertical` with 19.71 GB/s, which clearly outperforms the other implementations and reflects efficient register-level parallelism.

Interestingly, aligned and horizontal SIMD versions perform slightly worse than the vertical one, suggesting that memory alignment does not always translate into higher throughput in practice. Figure 2 and Table 2 show the transform benchmarks, which achieve significantly higher throughput overall.

Even the STL baseline version (`benchTransformStl`) reaches nearly 30 GB/s, indicating that this operation is already efficiently implemented and not severely bottlenecked by compute. Most transform implementations—including loop variants, STL-based, and OpenMP-assisted—cluster tightly around 29–30 GB/s, showing only minor variation.

The top performer is `XsimdTransform` at 30.53 GB/s, closely followed by `OmpSimdTransformRangeInnerLoop`. This confirms that transform is likely memory-bound, and further parallelization or vectorization provides only limited gains.

Overall, the reduce benchmark benefits more clearly from explicit SIMD optimization, while transform shows high performance even without it due to its inherent memory-access characteristics.

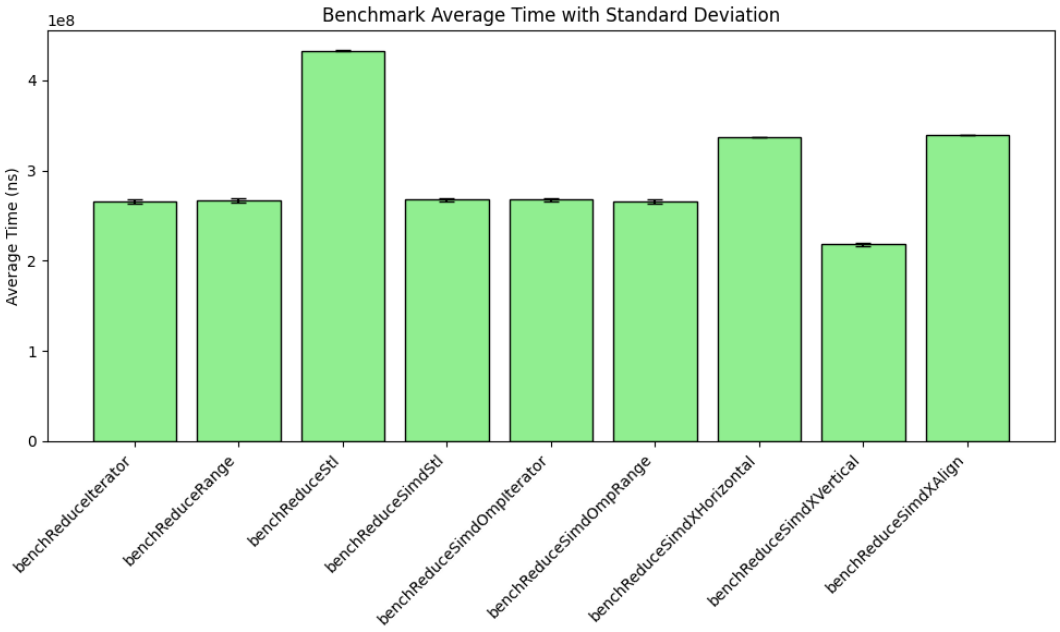

Figure 1: Reduce benchmark results (throughput vs. implementation)

Benchmark	**Throughput [GB/s]**
benchReduceStl	9.93
benchReduceSimdStl	16.05
benchReduceSimdXVertical	**19.71**
benchReduceSimdXHorizontal	12.75
benchReduceSimdXAlign	12.67
benchReduceOmpIterator	16.05
benchReduceOmpRange	16.14

Table 1: Throughput results of reduce benchmarks

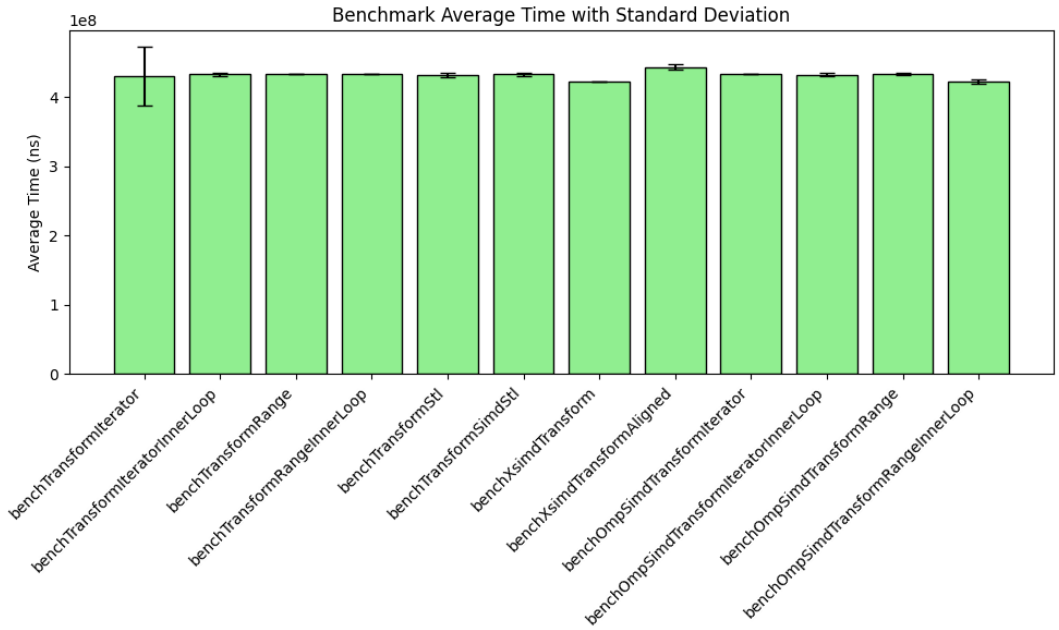

Figure 2: Transform benchmark results (throughput vs. implementation)

Benchmark	**Throughput [GB/s]**
benchTransformStl	29.82
benchTransformSimdStl	29.78
benchTransformRangeInnerLoop	29.75
benchTransformIteratorInnerLoop	29.77
benchOmpSimdTransformRangeInnerLoop	**30.51**
benchXsimdTransform	**30.53**
benchXsimdTransformAligned	29.10

Table 2: Throughput results of transform benchmarks

2.2 Adapting reduce and transform