PARALLELIZING THE SEQUENCE ALIGNMENT ALGORITHM (SMITH-WATERMAN ALGORITHM)

R. SHRI HARI PRIYA, SRINIDHI TARIGOPPULA (AI&DS-B)

- **Overview**:

  Sequence alignment algorithms are computational techniques used in bioinformatics to identify and measure the similarity between two or more biological sequences, such as DNA, RNA, or protein sequences.

  The primary goal of sequence alignment is to find the **optimal arrangement** of these sequences, highlighting regions of similarity and revealing potential evolutionary relationships.

- Diagonal values represent the scores of aligning subsequences of the input sequences. Parallelizing the computation of these diagonal values can significantly improve the overall performance.
- Divide the computation of diagonal values among multiple processors or threads.
- Each processor or thread can be assigned the responsibility for computing the values along a specific diagonal of the matrix.

- **Limitations of Traditional Computing**:
- Traditional computing methods face significant challenges when dealing with the computational demands of the Smith-Waterman Algorithm

- The algorithm's dynamic programming approach requires computing a similarity matrix for all possible alignments between two sequences, resulting in a time complexity of O(n*m), where n and m are the lengths of the sequences being compared.

- As a result, the computational resources required grow rapidly with the size of the sequences, making it impractical to analyse large-scale genomic datasets using traditional computing resources.

- **Advantages of High-Performance Computing (HPC)**:
  - **Parallel Processing**: HPC systems are designed to efficiently handle parallel tasks by utilizing multiple processors or nodes simultaneously. This parallelization allows the algorithm to be divided into smaller tasks that can be executed concurrently, significantly reducing the overall computation time.

- **Task Distribution**: HPC systems can distribute computation tasks across multiple processors or nodes, allowing for efficient utilization of computing resources. This distribution of tasks enables the algorithm to scale effectively, even when processing large datasets.

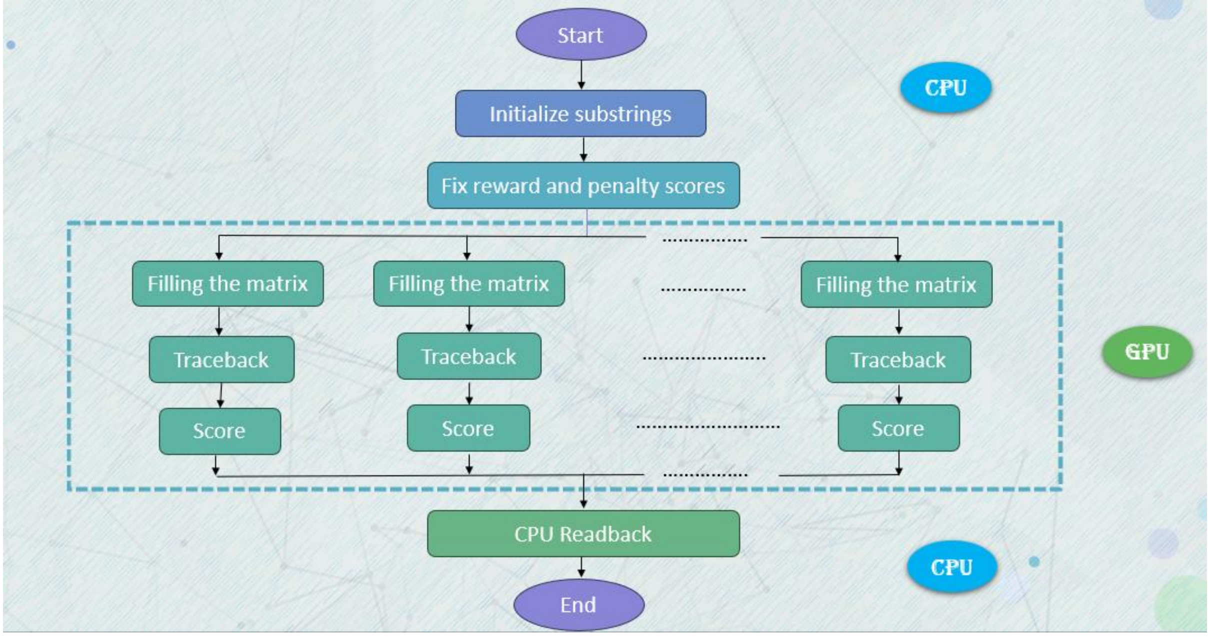
- **Reduced Computation Time**: By harnessing the parallel processing capabilities of HPC systems, the time required to perform sequence alignment using the Smith-Waterman Algorithm can be drastically reduced, making it feasible to analyse large-scale genomic datasets in a reasonable amount of time.

- **Aspects of Parallelism**:
  1. **Data Parallelism** - Divide the alignment process into independent chunks that can be executed concurrently. This can be particularly effective when dealing with a large number of sequences or long sequences.

  2. **Matrix Computation** - The computation of the scoring matrix involves calculating scores for each cell, based on the values of neighboring cells. Parallelize the computation of matrix cells by dividing the matrix into smaller submatrices. Each processor or thread can be responsible for computing scores within its assigned submatrix.

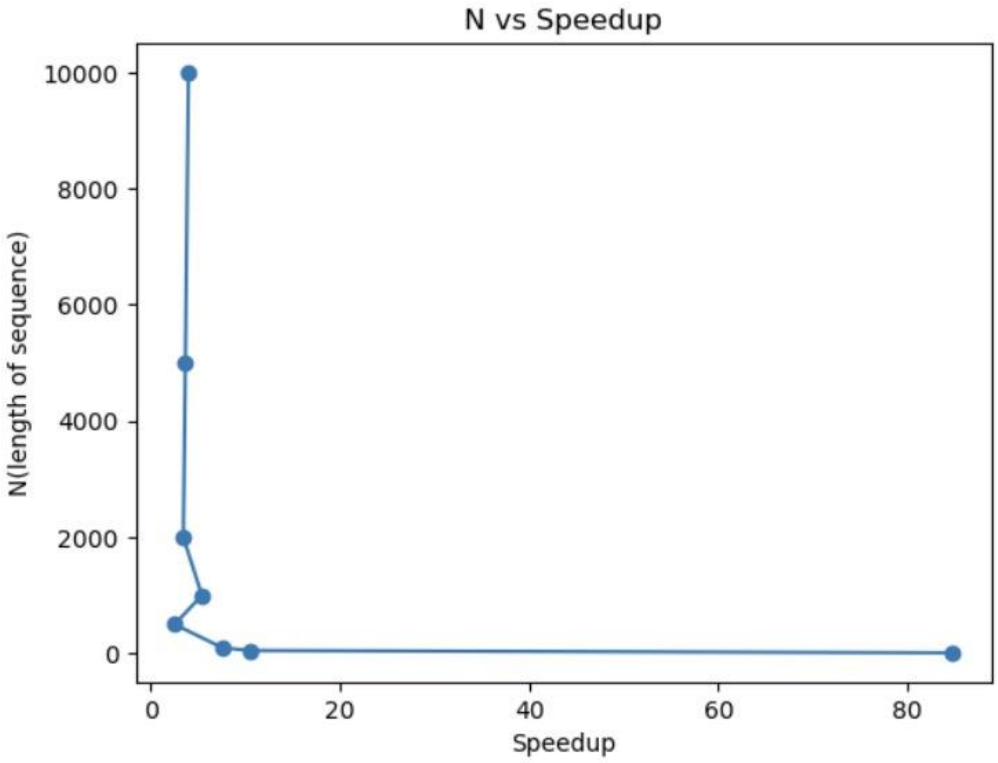
  3. **Parallel Traceback Paths** - If there are multiple optimal alignment paths or if alternative alignments need to be explored, backtracking can be parallelized along different paths. Each processor or thread can explore a different path independently, possibly leading to different optimal alignments.

- **Block Diagram & Architecture**:

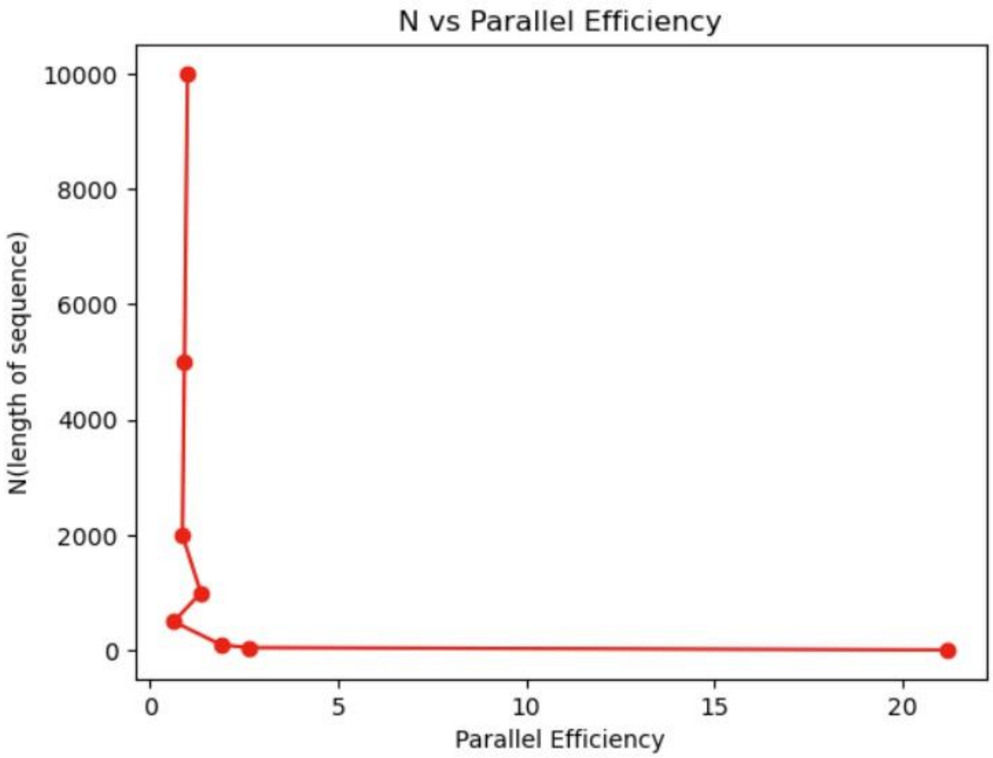


- **Challenges**:
  Ensuring balance between efficient parallelization strategies, memory management and synchronization so that the best level of optimization is achieved.

- **Performance Metrics**:

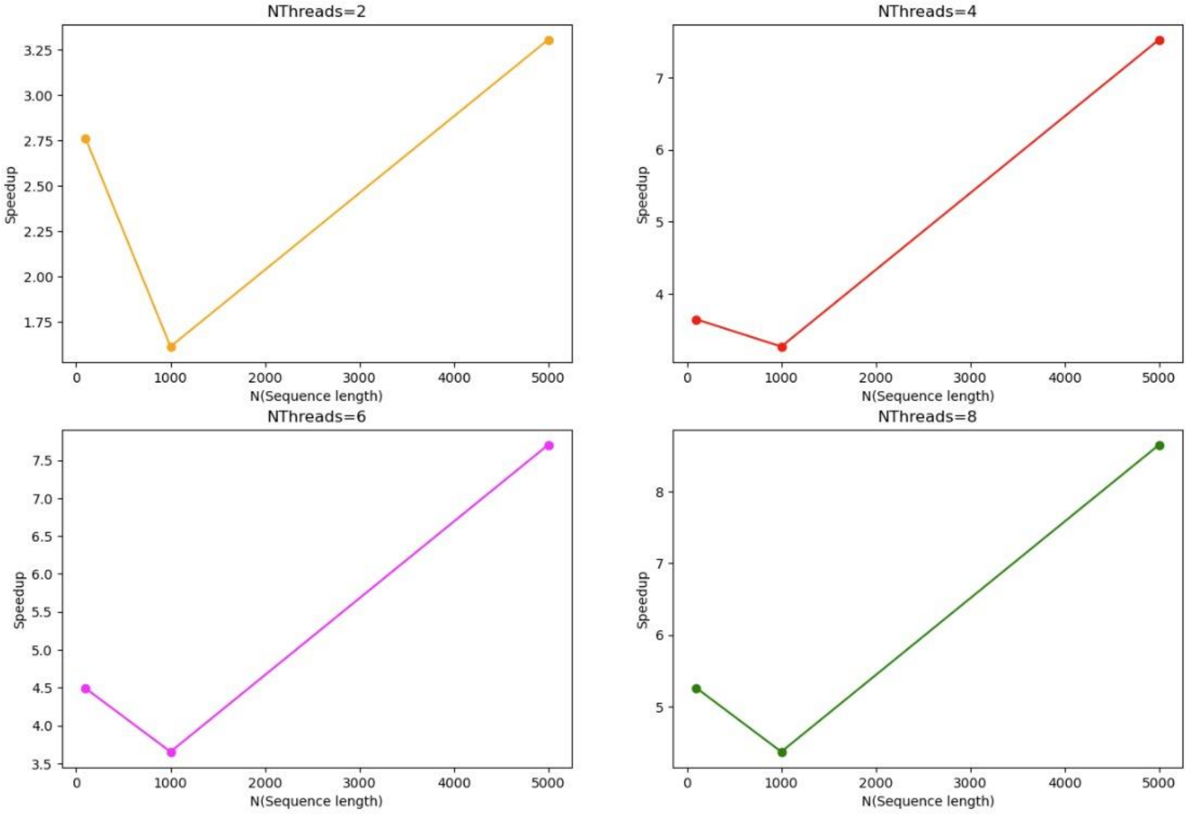


N (Sequence Length)	Speed Up
10	84.75
50	10.41071
100	7.525114
500	2.510253
1000	5.391587
2000	3.430781
5000	3.645732
10000	3.972967

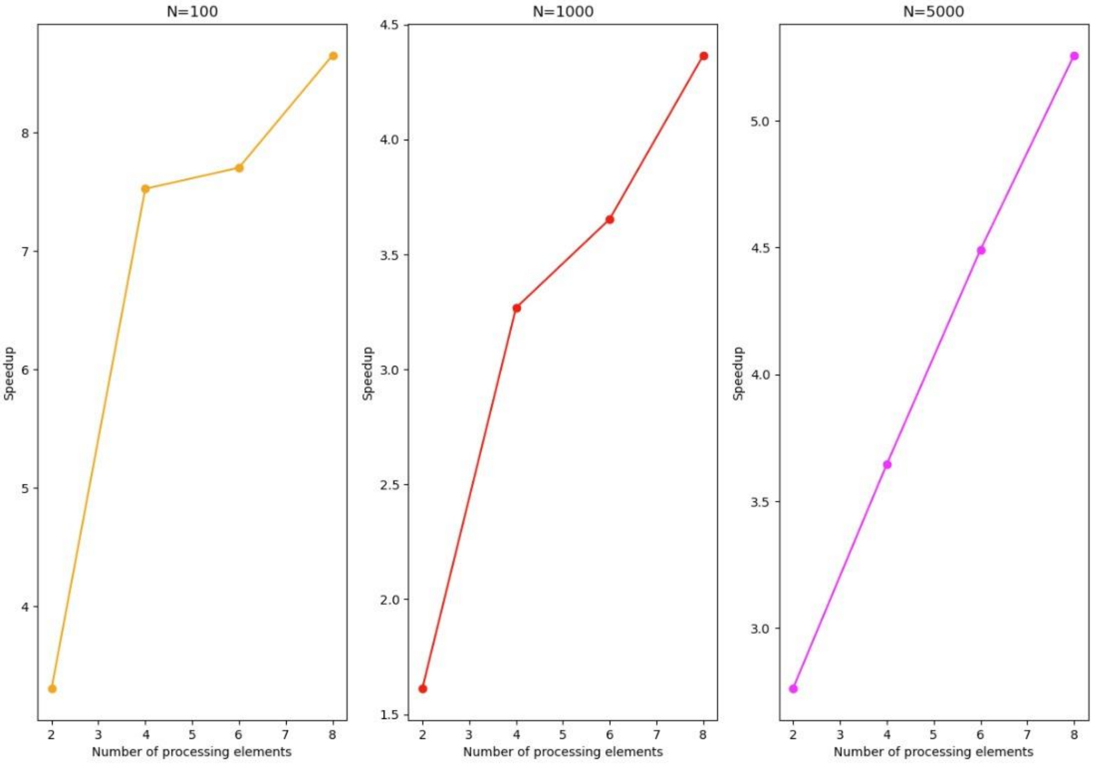


N (Sequence Length)	Speed Up
10	84.75
50	10.41071
100	7.525114
500	2.510253
1000	5.391587
2000	3.430781
5000	3.645732
10000	3.972967

Speed Up vs N



Speed Up vs No. of PE


N=100
Speedup
Number of processing elements
N=1000
Speedup
Number of processing elements
N=5000
Speedup
Number of processing elements