SMITH-WATERMAN ALGORTIHM

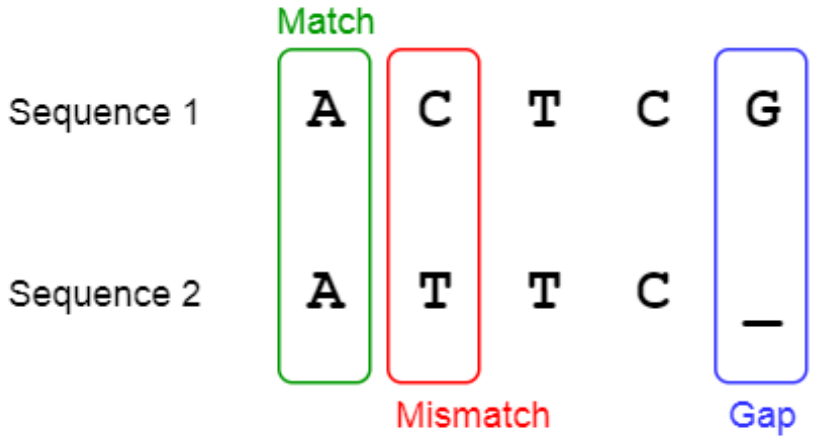
Sampreeth Reddy Seelam

CSE 708 (Dr. Russ Miller)

University at Buffalo The State University of New York

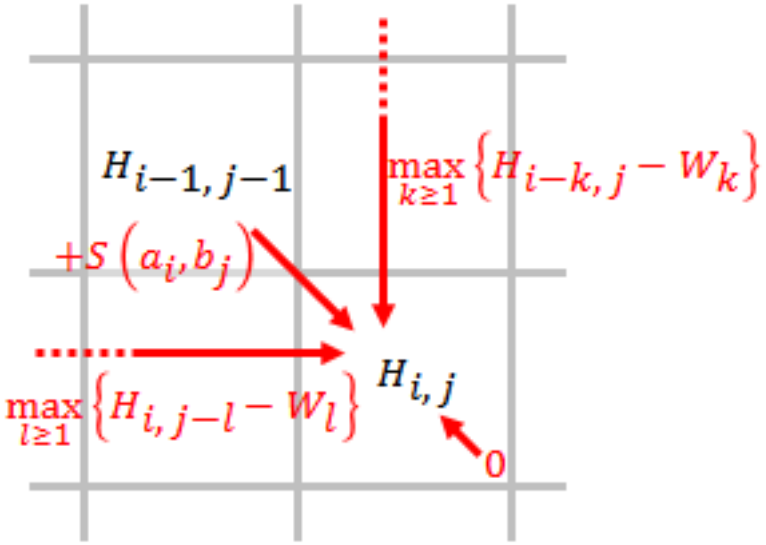
Smith-Waterman Algorithm

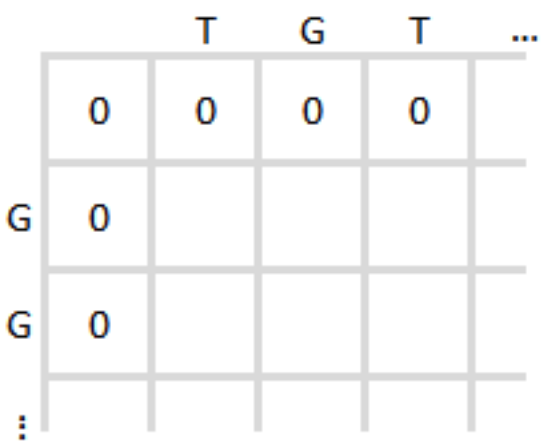
- Performs local sequence alignment.
- For determining similar regions between two strings of nucleic acid sequences or protein sequences.
- Dynamic programming algorithm that is guaranteed to find local alignment.
- Compared to Needleman-Wunsch algorithm, negative scores are set to zero.
- Time complexity of the algorithm is O(mn).

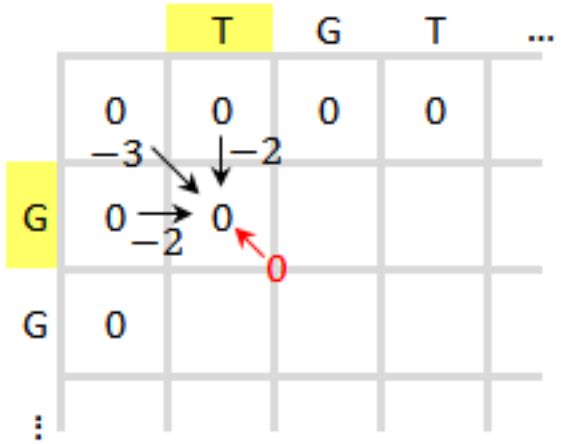


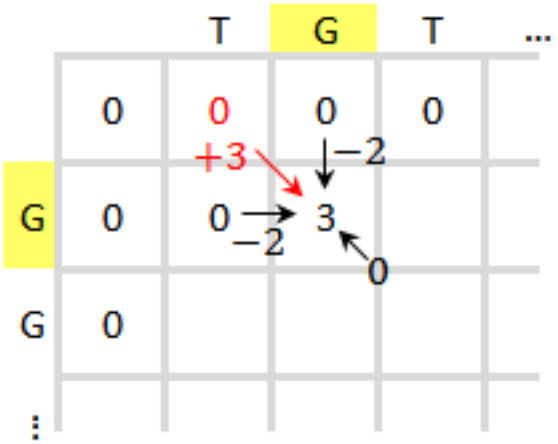
Let and

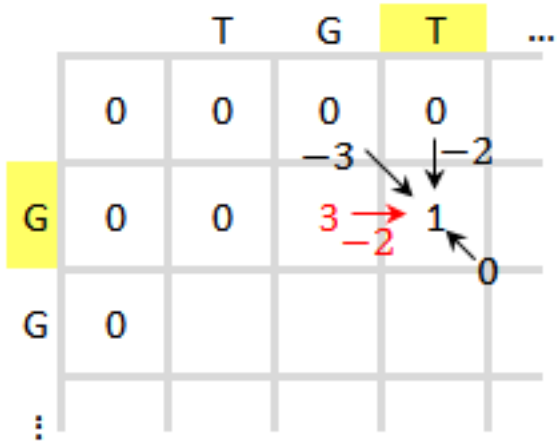
Algorithm





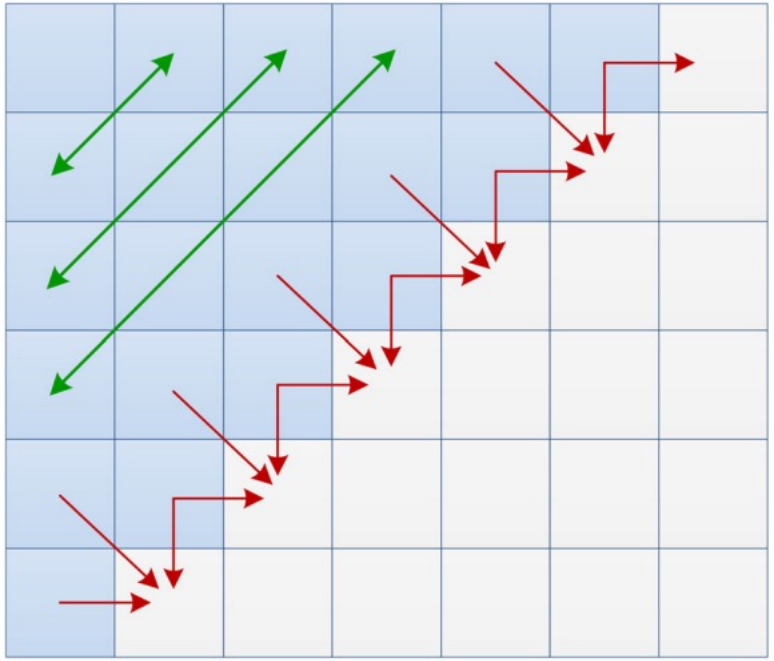




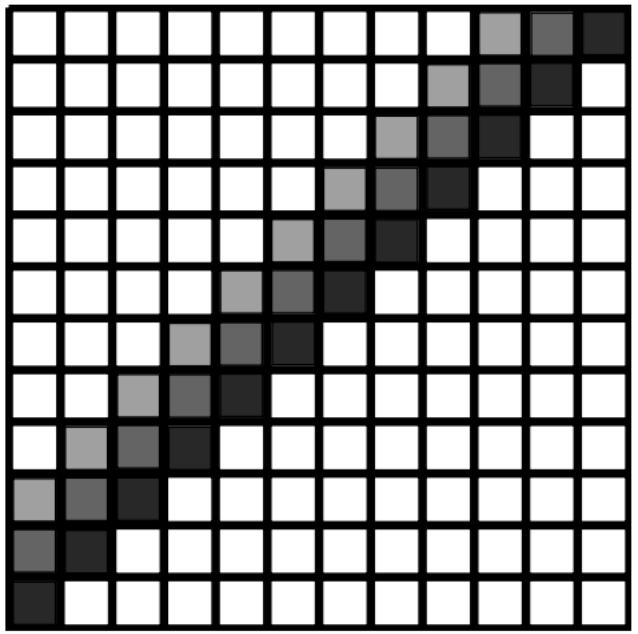


Parallel Implementation

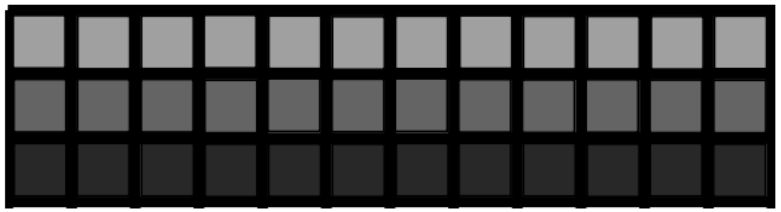
- Each cell of the matrix is dependent on 3 cell computations(Red Arrows)
- Anti-Diagonal cells are independent of each other.
- Compute them in blocks parallelly .



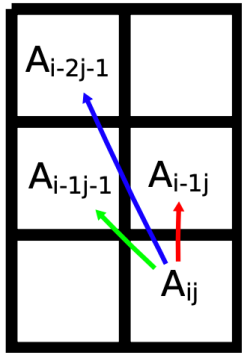
Parallel Implementation



Computation of the Score Matrix: the black anti-diagonal is calculated using two light shaded anti-diagonals



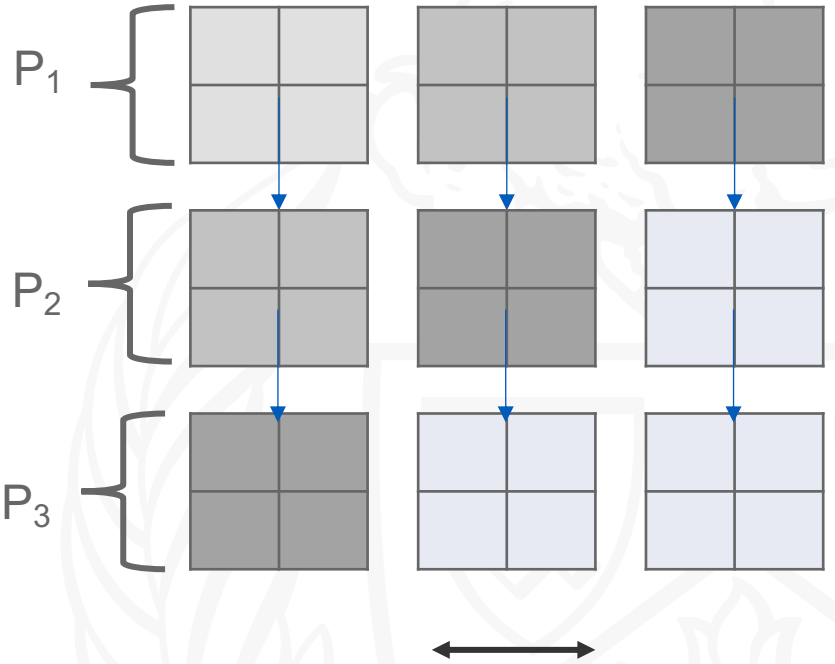
The entire score matrix is transformed into three rows. Each row of the same colour is equal to the same colour anti-diagonal in left figure.



The score matrix after transformation and the entries involved in computing a score matrix entry

Modification for MPI

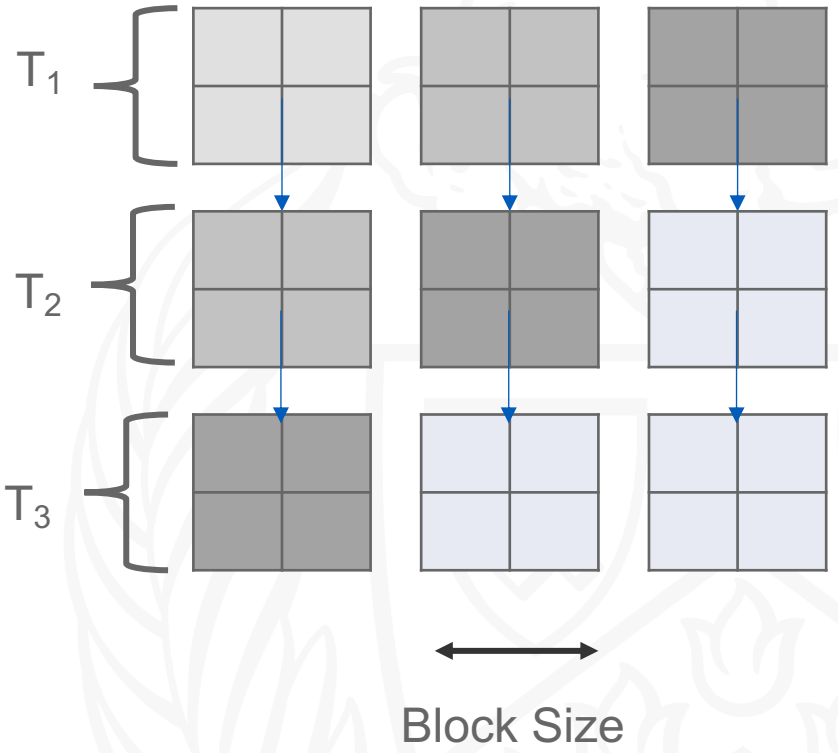
- Instead of computing each cell, divide the the matrix into blocks.
- Rows are divided based on number of processors.(Query Sequence).
- Columns are divided based on a block size. The block size will decide how many cells should be computed in a single iteration.
- Once the blocks are computed they are transferred to the next processor in line and the current processor continues to compute the next block.



The blocks are computed in increasing shade(from lighter to darker).

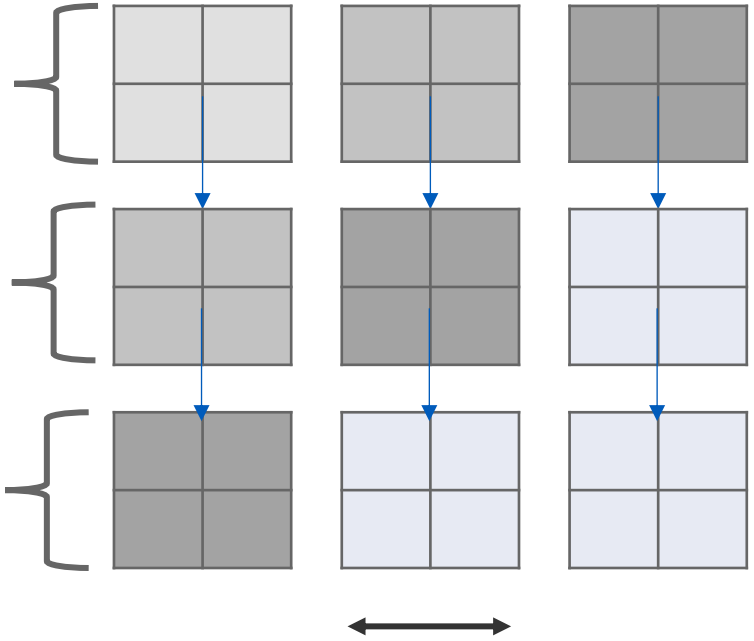
Does this work for OpenMP?

- No, Because the the string is split in case of MPI and the other nodes wait till the pervious node sends a message to start the computation.
- Threads do not know when to start and when to signal other threads to start.



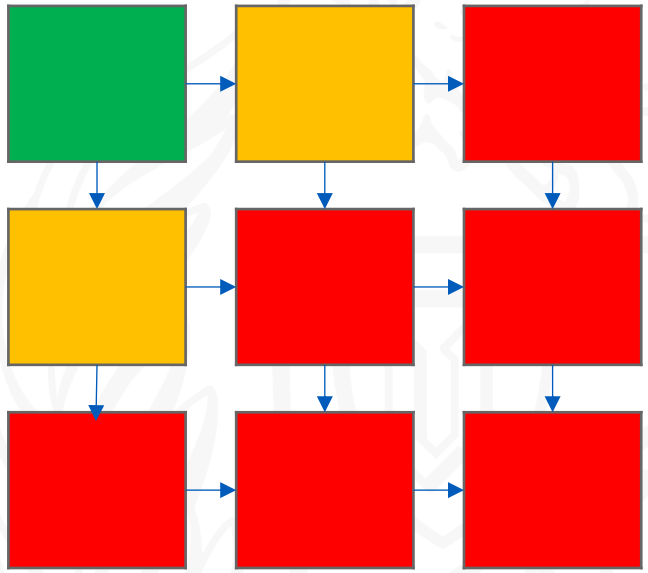
The blocks are computed in increasing shade(from lighter to darker).

Solution for OpenMP



Block Size

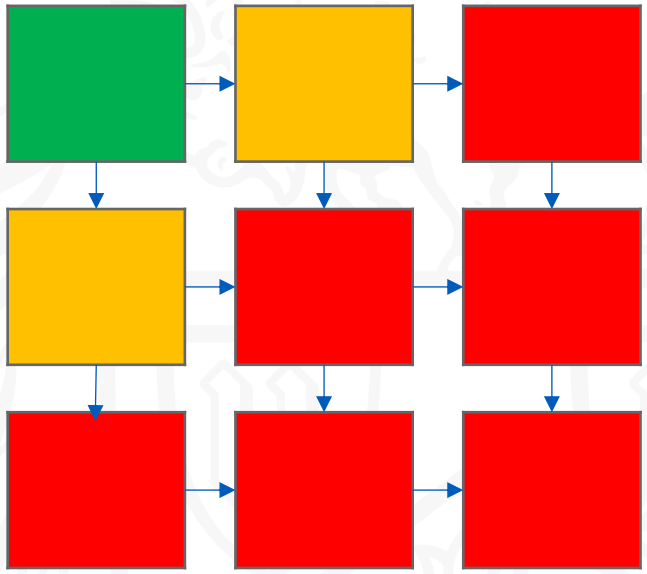
The blocks are computed in increasing shade(from lighter to darker).



Maintain 3 states for each block.

Solution for OpenMP

- 3 states for each block.
  - NOT STARTED
  - IN PROGRESS
  - DONE
- Initially start with a single thread and work on first block.
- Once done, create a task for the left and bottom block if already not started.
- After the completion of all the blocks identify the maximum and traceback.

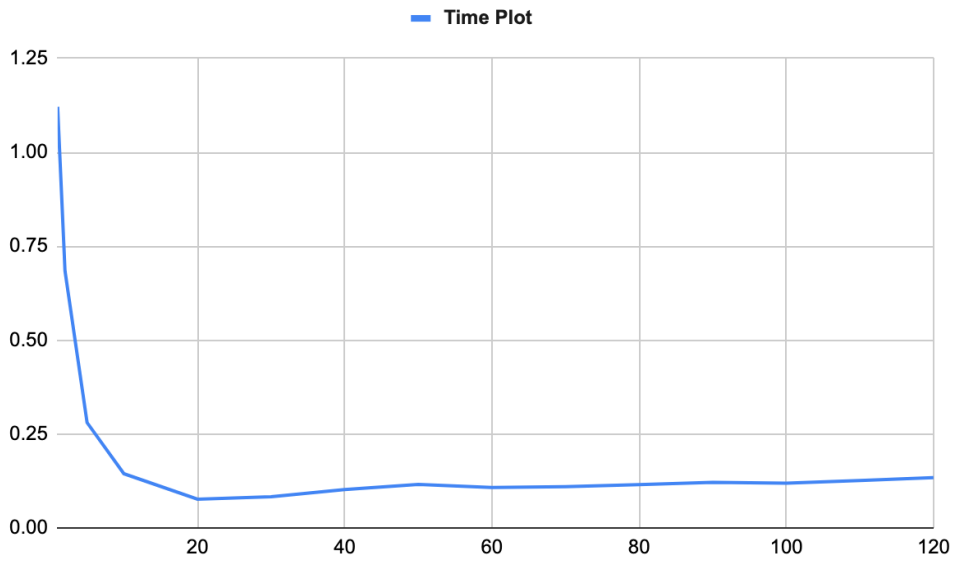


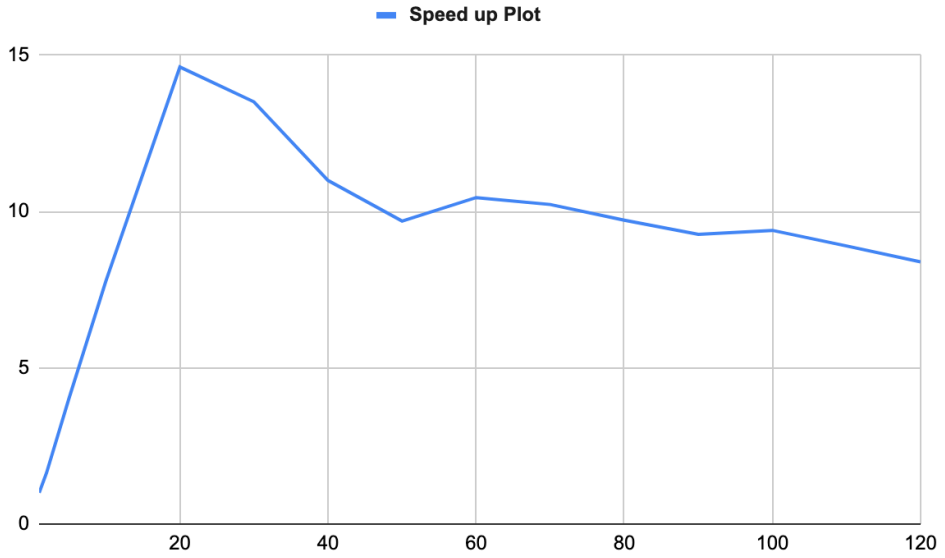
Maintain 3 states for each block.

Dataset: N=1000

Threads	Time
1	1.12
2	0.68
5	0.28
10	0.14
20	0.08
30	0.08
40	0.10
50	0.12

Threads	Time
60	0.11
70	0.11
80	0.12
90	0.12
100	0.12
110	0.13
120	0.13

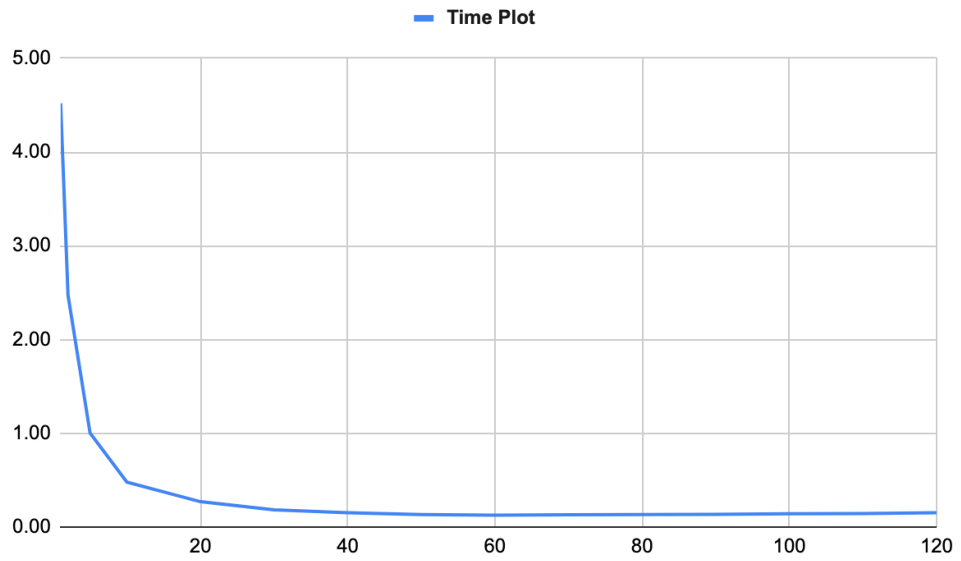


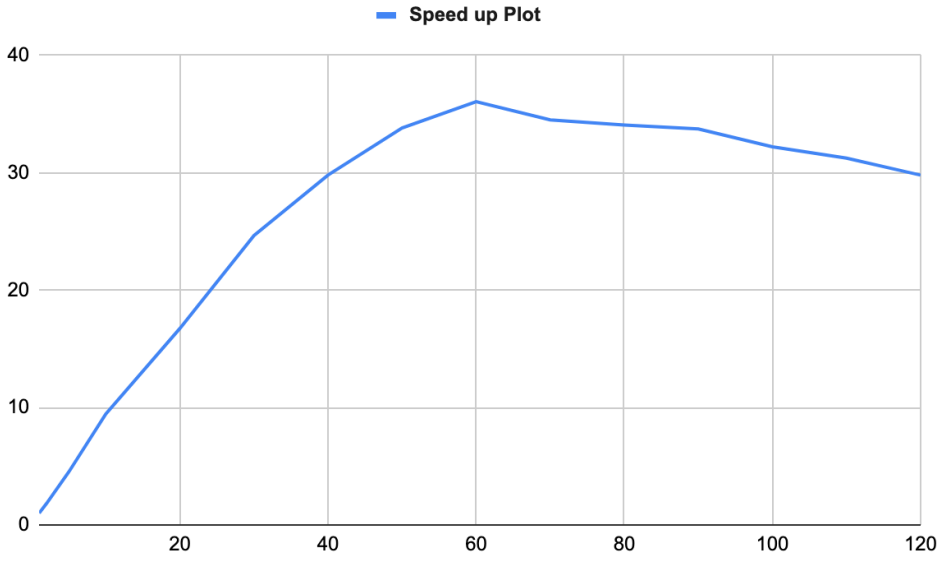


Dataset: N=2000

Threads	Time
1	4.52
2	2.47
5	1.00
10	0.48
20	0.27
30	0.18
40	0.15
50	0.13

Threads	Time
60	0.13
70	0.13
80	0.13
90	0.13
100	0.14
110	0.14
120	0.15

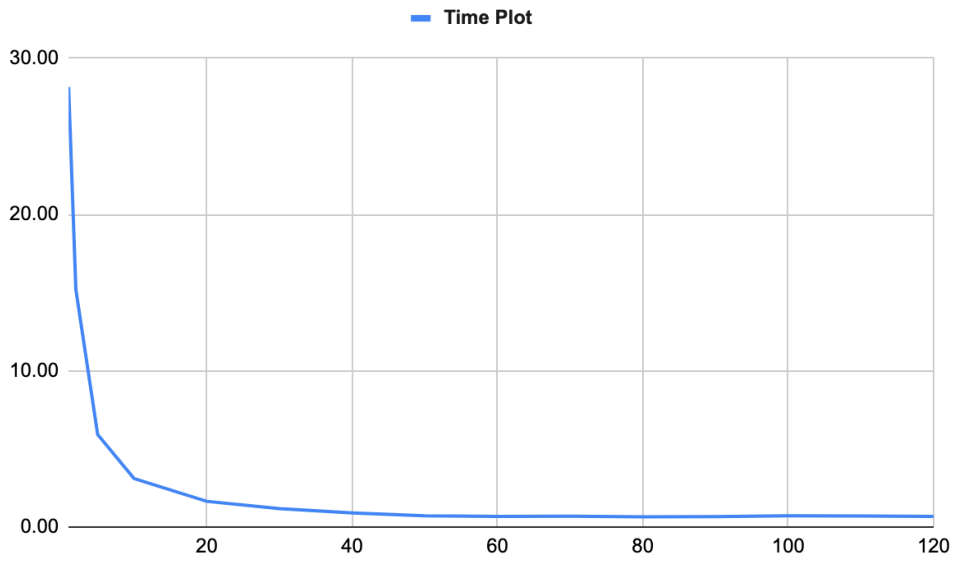


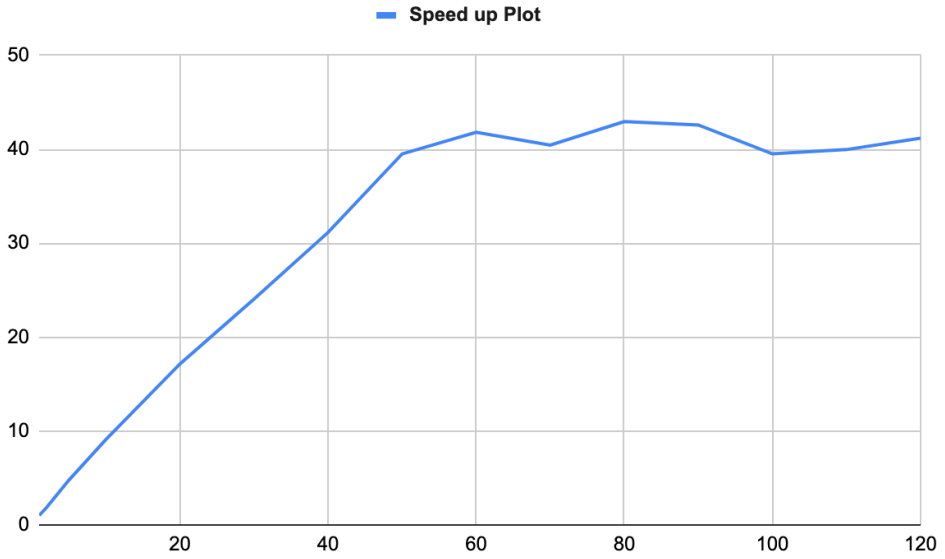


Dataset: N=5000

Threads	Time
1	28.17
2	15.21
5	5.92
10	3.10
20	1.64
30	1.17
40	0.90
50	0.71

Threads	Time
60	0.67
70	0.70
80	0.66
90	0.66
100	0.71
110	0.70
120	0.68

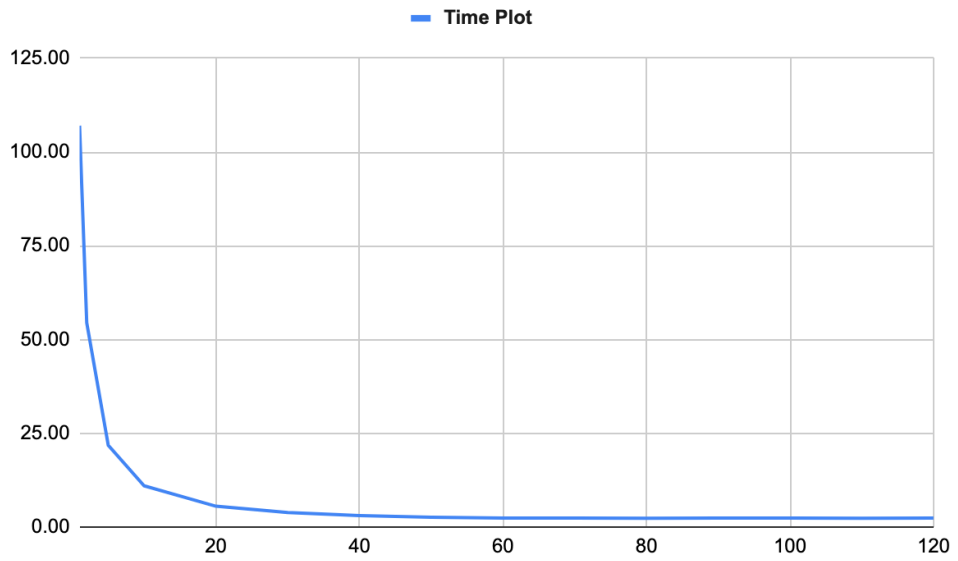


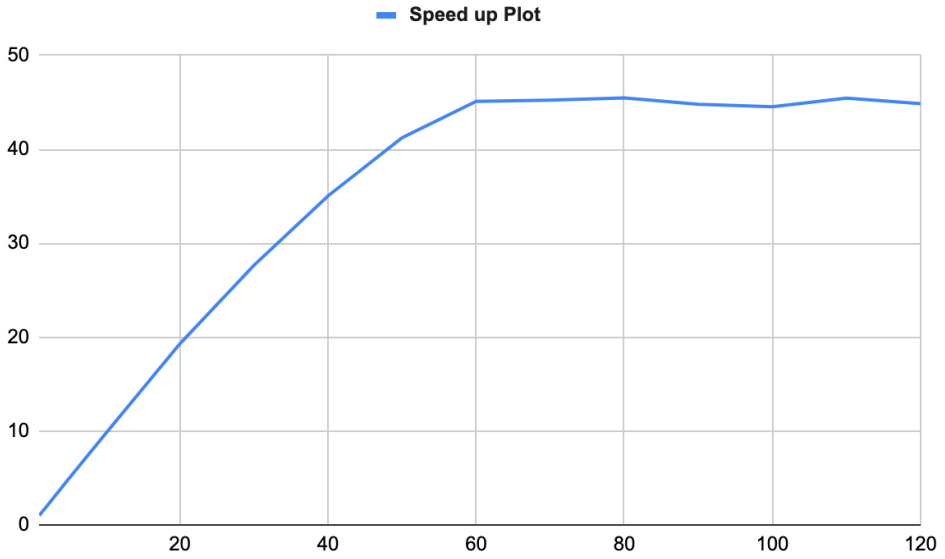


Dataset: N=10000

Threads	Time
1	107.03
2	54.47
5	21.80
10	10.97
20	5.55
30	3.87
40	3.06
50	2.60

Threads	Time
60	2.37
70	2.37
80	2.35
90	2.39
100	2.40
110	2.36
120	2.39





Conclusions

- Similar approach as MPI has been used.
- Avoided race conditions by using additional matrix to identify which blocks are completed and which blocks are in progress.
- Large Dataset has more scalability than smaller dataset.
- Optimal point for the large dataset was not identified due to limitation on number of processer per node.
- OpenMP is faster than MPI as there is no communication time involved in case of OpenMP.
- For larger dataset than 10000, we need to use the MPI approach as the shared memory will not be sufficient identify the optimal alignment as the matrix increases with power of 2.

References

- https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm
- https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Jian-Chen-Spring-2019.pdf (For Images)
- Parallelizing the Smith-Waterman Algorithm using OpenSHMEM and MPI-3 One-Sided Interfaces - Matthew Baker, Aaron Welch, Manjunath Gorentla Venkata.

Thank You