SMITH-WATERMAN ALGORITHM

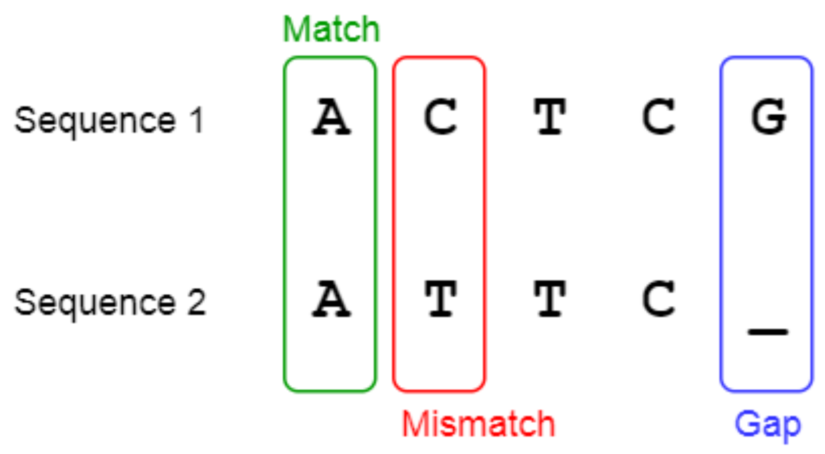
Sampreeth Reddy Seelam

CSE 633 Parallel Algorithms (Dr. Russ Miller)

University at Buffalo The State University of New York

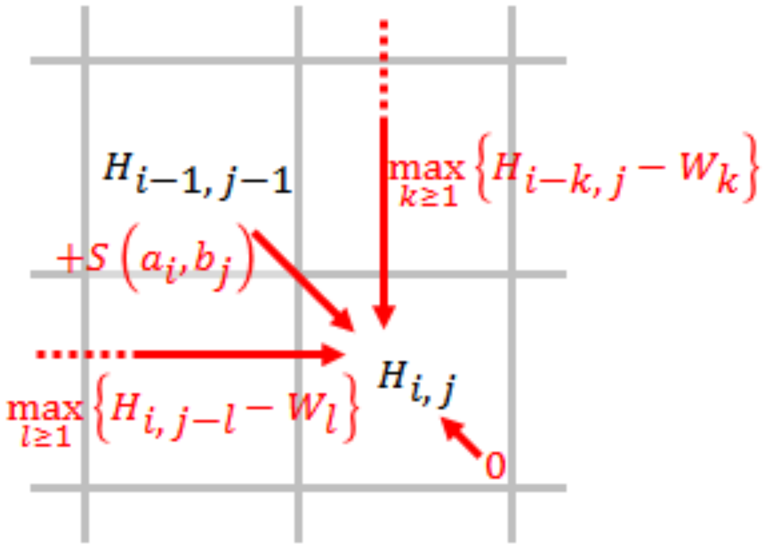
Smith-Waterman

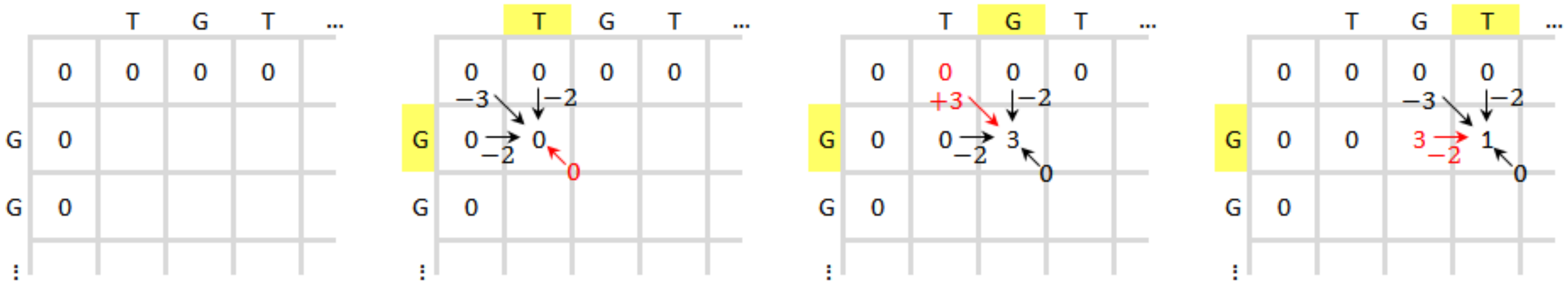
- performs local sequence alignment.
- for determining similar regions between two strings of nucleic acid sequences or protein sequences.
- Dynamic programming algorithm that is guaranteed to find local alignment.
- Compared to Needleman-Wunsch algorithm, negative scores are set to zero.
- Time complexity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



Let and

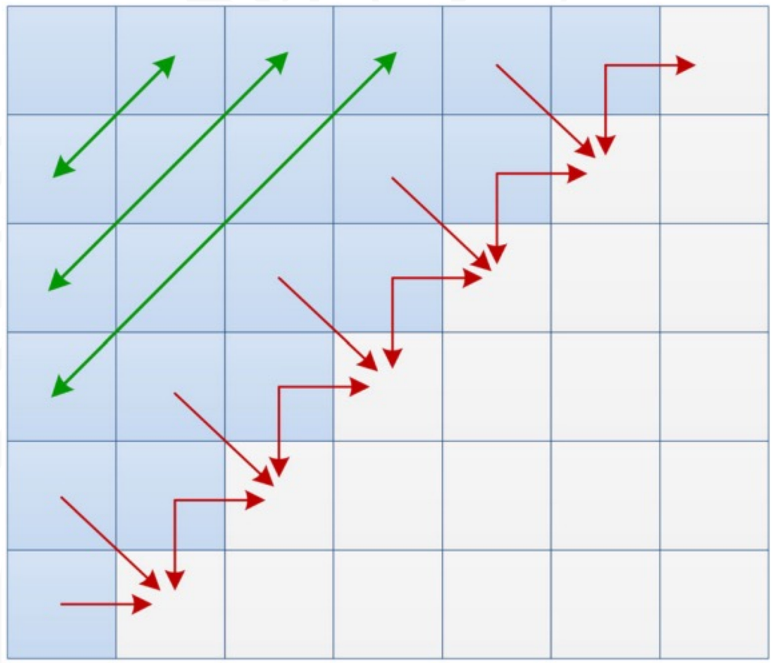
Algorithm


$H_{i-1,\,j-1}$
$\max_{k\geq 1}\{H_{i-k,\,j} - W_k\}$
$+S(a_i, b_j)$
$H_{i,\,j}$
$\max_{l\geq 1}\{H_{i,\,j-l} - W_l\}$
0

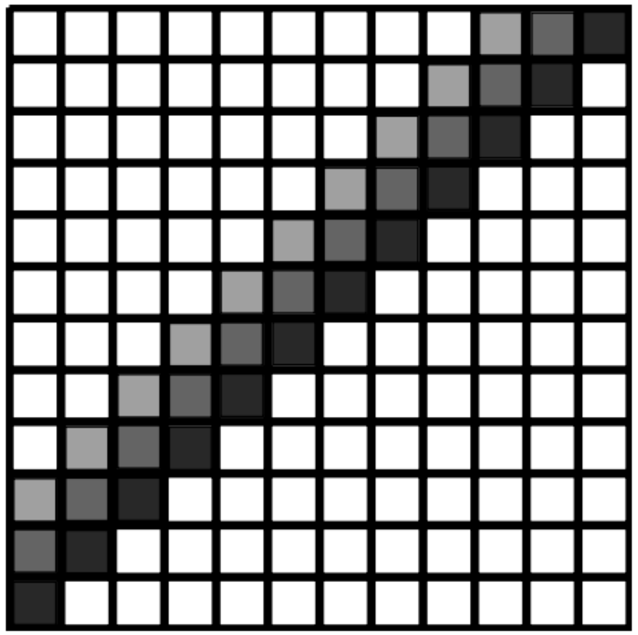


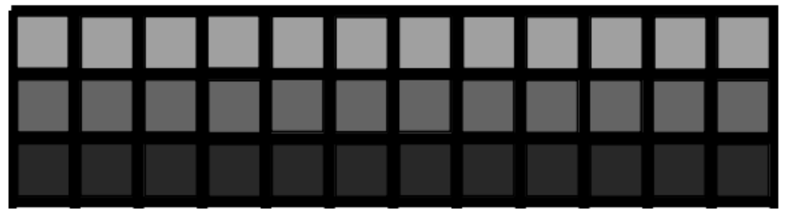
Parallel Implementation

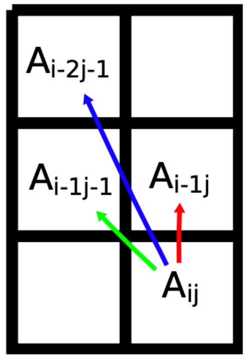
- Each cell of the matrix is dependent on 3 cell computations(Red Arrows)
- Anti-Diagonal cells are independent of each other.
- Compute them in blocks parallelly .



Parallel Implementation


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

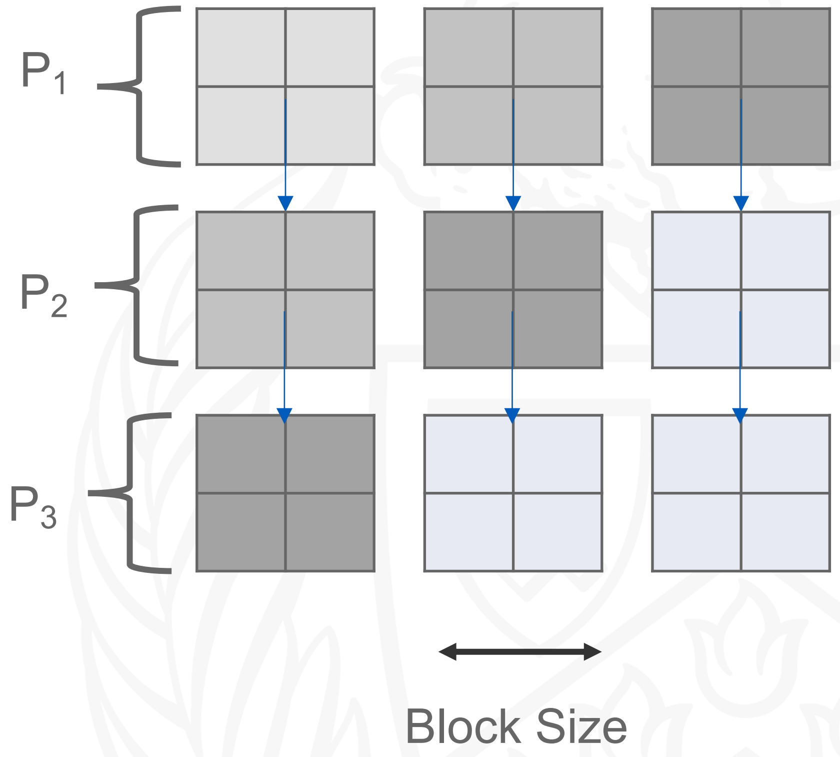

The entire score matrix is transformed into three rows. Each row of the same colour is equal to the same colour anti-diagonal in left figure.



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

Modification

- 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.

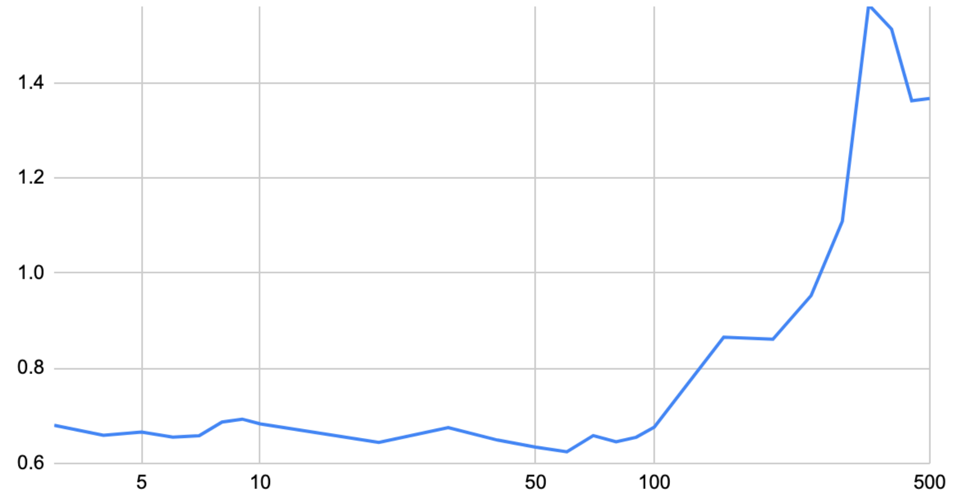


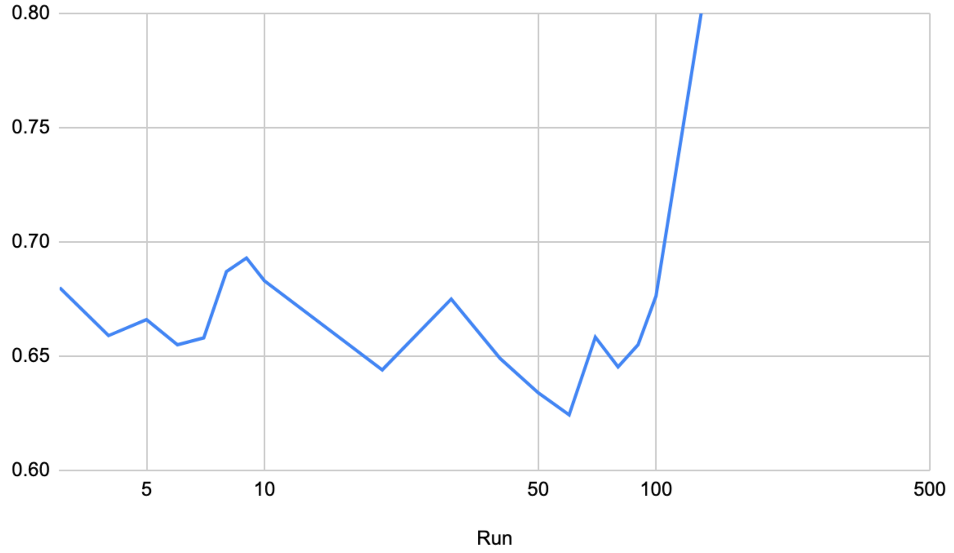
The blocks are computed in increasing shade(from lighter to darker).

Choosing Optimal Block Size (N=1000, P=5)

Block Size	Time
1	0.663
2	0.654
5	0.66
7	0.658
10	0.683
20	0.644
30	0.675
40	0.649
50	0.634
60	0.624
70	0.658

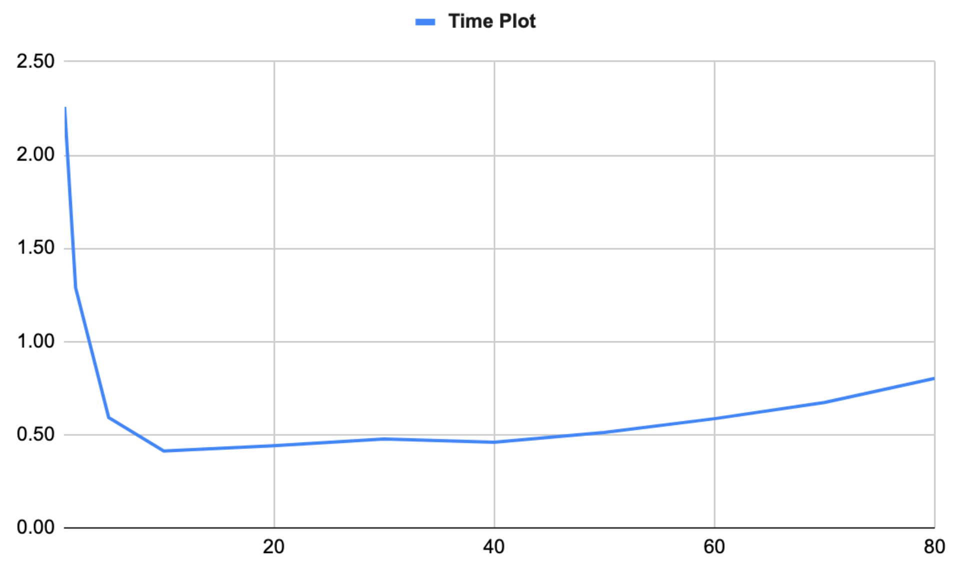
Block Size	Time
80	0.645
90	0.655
100	0.676
150	0.865
200	0.861
250	0.952
300	1.1
350	1.56
400	1.51
450	1.32
500	1.36

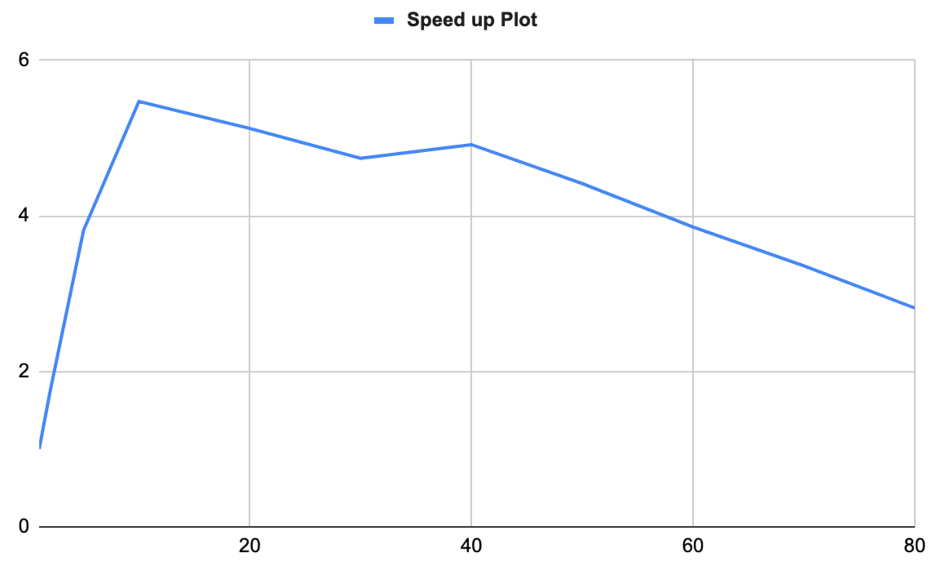




Dataset : N = 1000

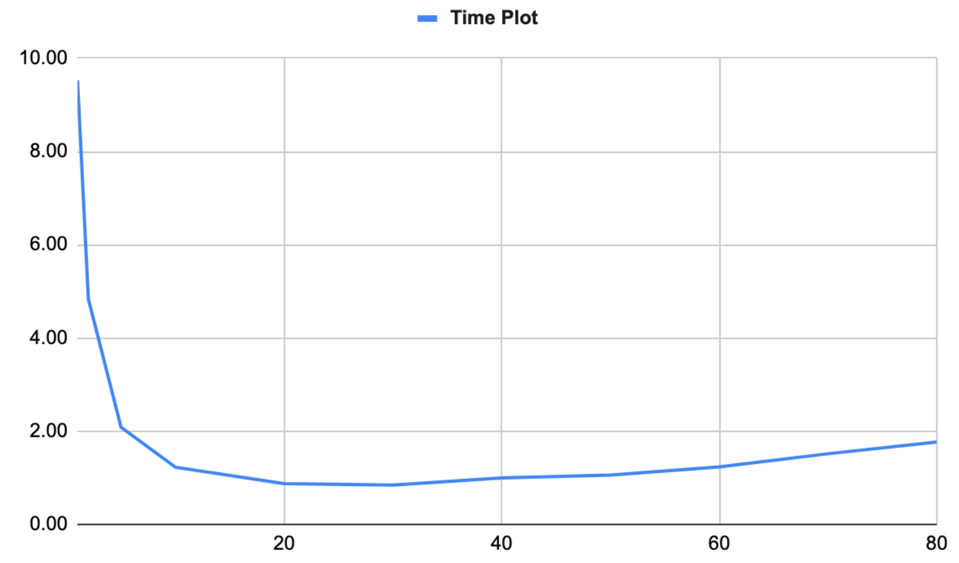
Processors	Time
1	2.26
2	1.29
5	0.59
10	0.41
20	0.44
30	0.48
40	0.46
50	0.51
60	0.59
70	0.67
80	0.80

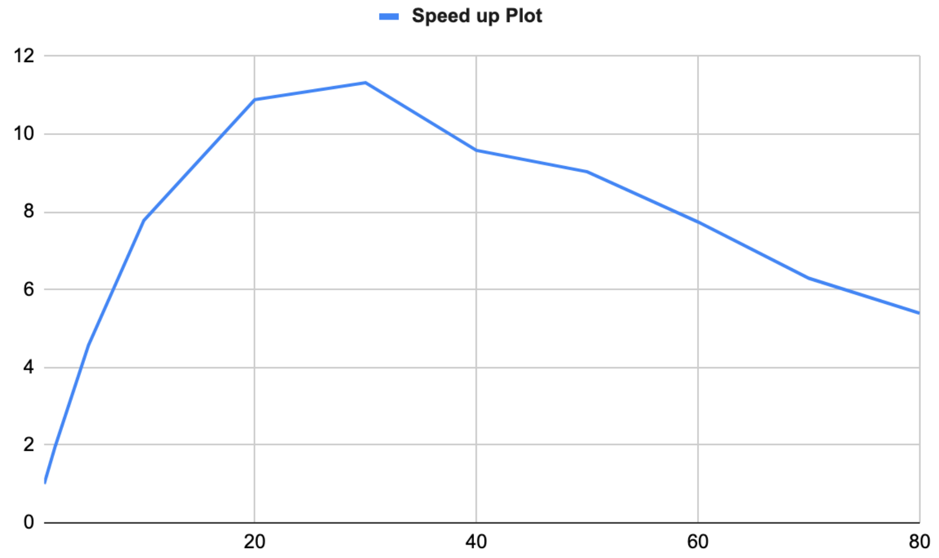




Dataset : N = 2000

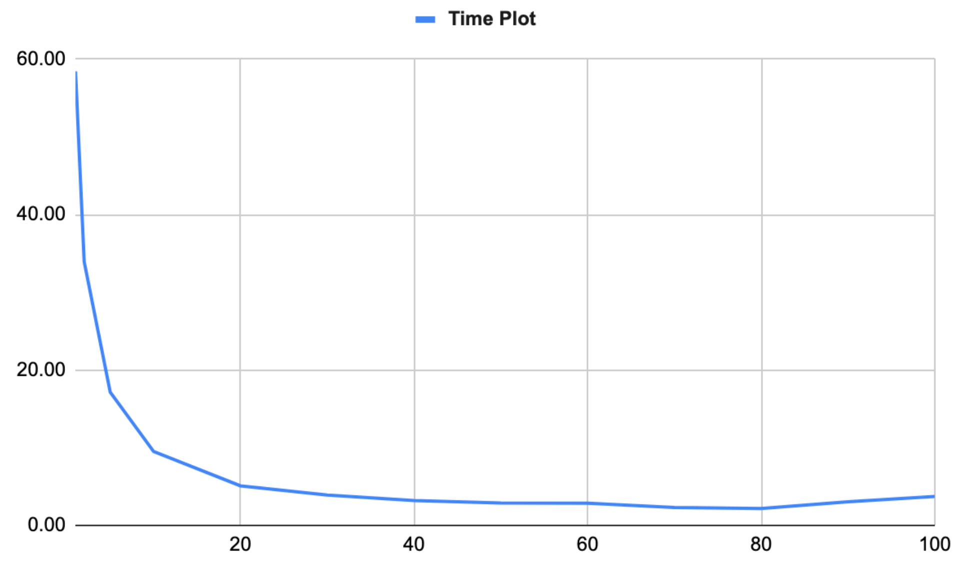
Processors	Time
1	9.53
2	4.83
5	2.09
10	1.23
20	0.88
30	0.84
40	1.00
50	1.06
60	1.23
70	1.52
80	1.77

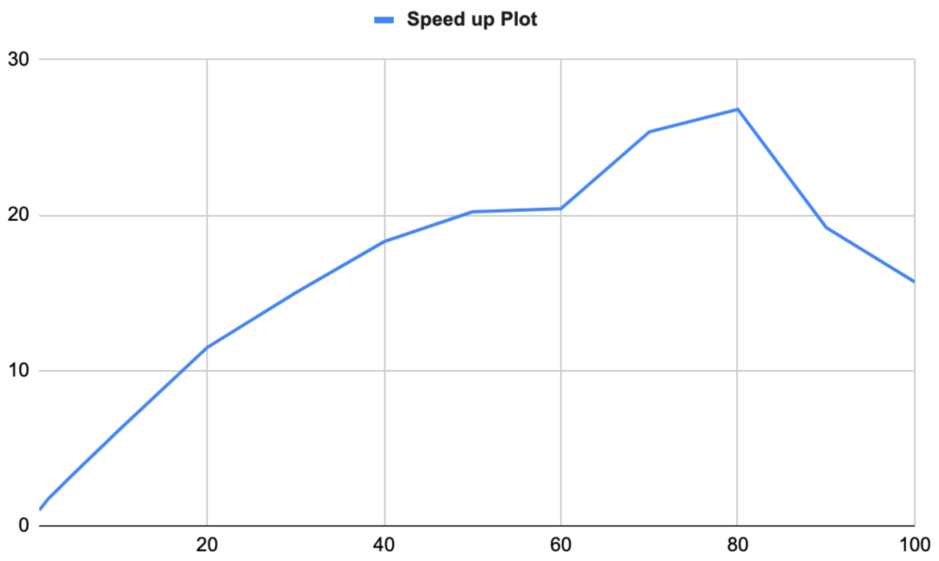




Dataset : N = 5000

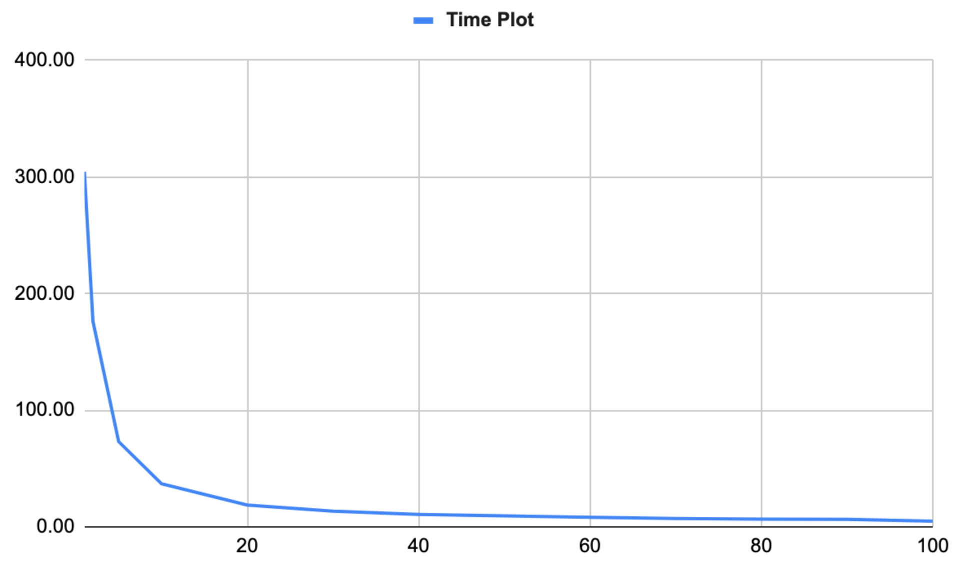
Processors	Time
1	58.45
2	33.95
5	17.15
10	9.50
20	5.09
30	3.89
40	3.20
50	2.89
60	2.86
70	2.30
80	2.18
90	3.04
100	3.72

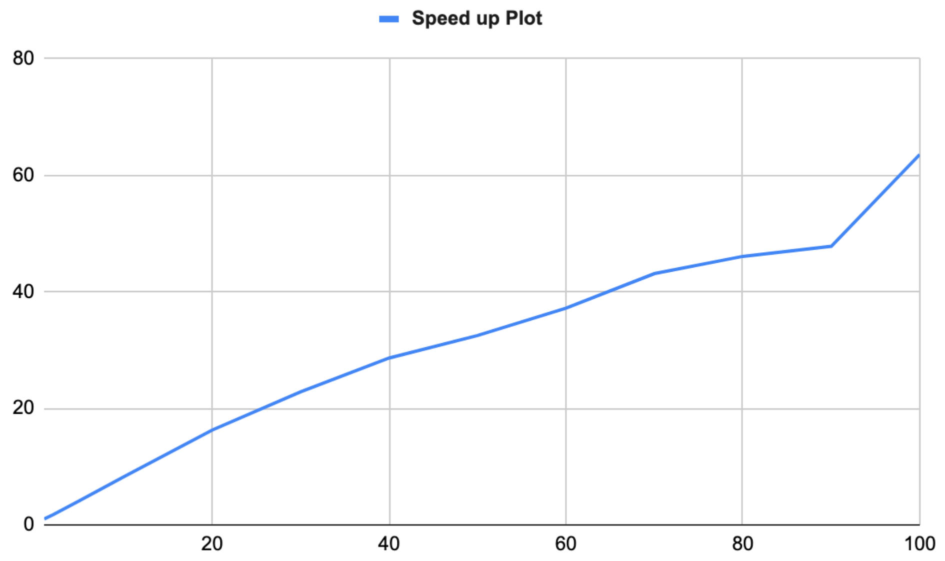




Dataset : N = 10000

Processors	Time
1	304.47
2	176.08
5	73.05
10	36.85
20	18.71
30	13.34
40	10.63
50	9.37
60	8.19
70	7.06
80	6.61
90	6.37
100	4.79





Conclusion

- Independent Task have been identified. Anti-Diagonals can be computed parallelly.
- Optimal Block size is around 5% of the data size.
- Large Dataset needs more processors to run faster.
- Optimal Point of processors increase with increase in dataset size.
- Memory required for calculating the matrix with increase with a power of 2. Hence it will become difficult to calculate matrix for large dataset.

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