SMITH-WATERMAN ALGORITHM OPENMP

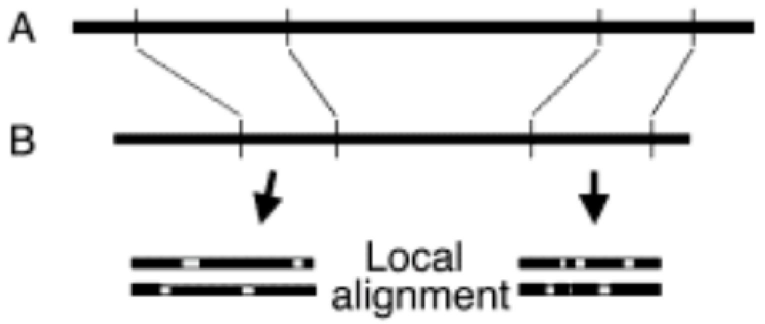
Sai Ram Gurram
CSE 633 (Dr. Russ Miller)

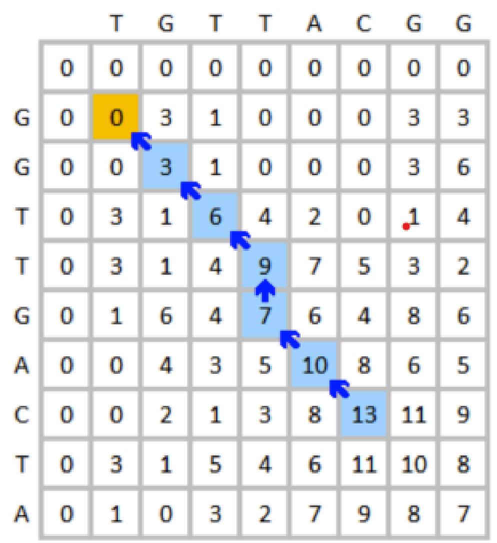
University at Buffalo The State University of New York

Smith-Waterman Algorithm

- The Smith-Waterman algorithm is a well-known method used in bioinformatics for local sequence alignment; that is, for aligning subsequences of proteins or nucleic acids.
- The Smith-Waterman algorithm is a dynamic programming algorithm used for local sequence alignment.
- It compares segments of all possible lengths and identifies regions of high similarity
- The main part of this algorithm is finding the scoring matrix of size m*n, so the time complexity of this algorithm is O(mn)., m,n are the lengths of the protein sequences

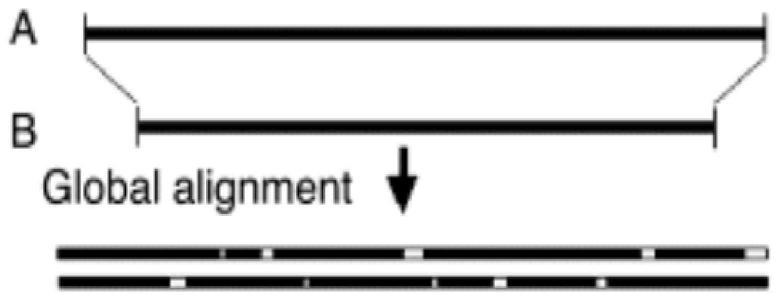
SMITH WATERMAN

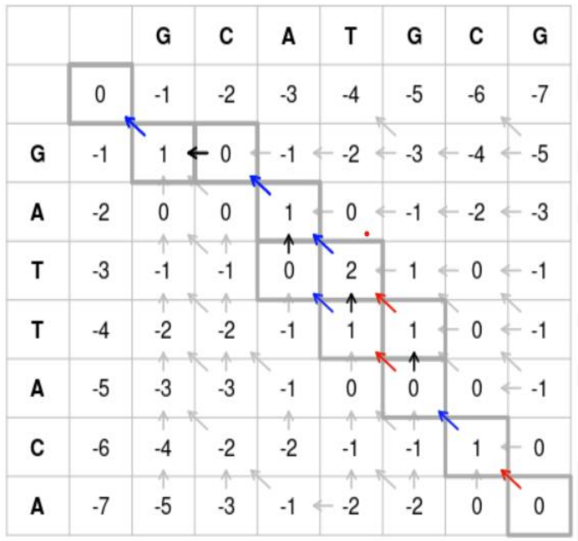




		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

NEEDLEMAN–WUNSCH





		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Algorithm

- Consider the maximum value from 4 cases
- Apply Gap penalty to top and left
- Match/Mismatch score to the diagonal value
- Set the score to zero if it is negative

Match/Mismatch: +3/-3

Gap penalty: -2

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k \ge 1}\{H_{i-k,j} - W_k\}, \\ \max_{l \ge 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \qquad (1 \le i \le n, 1 \le j \le m)$$

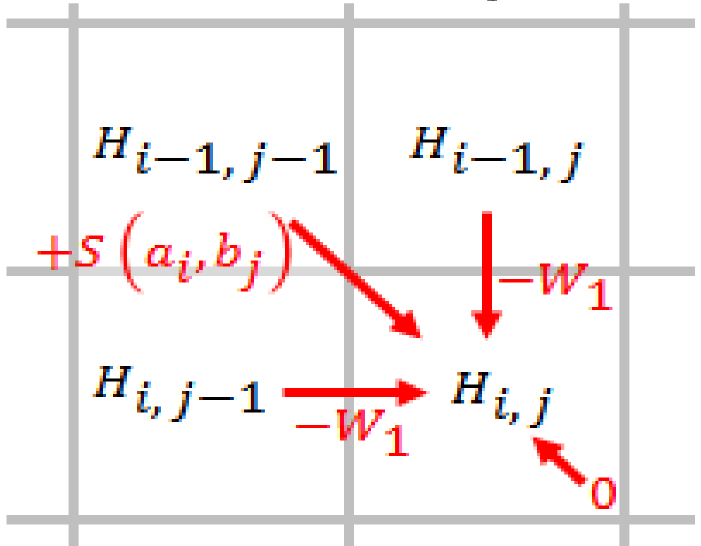
where

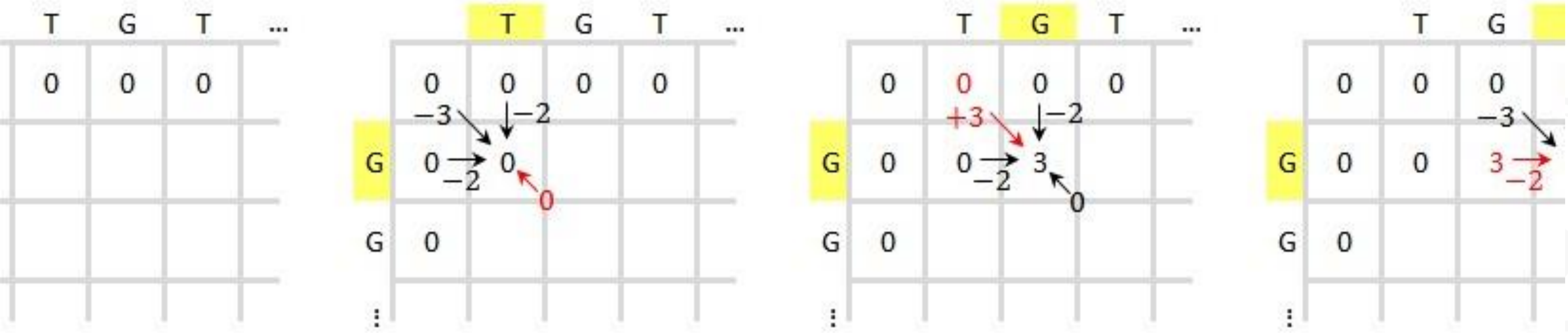
$H_{i-1,j-1} + s(a_i, b_j)$ is the score of aligning a_i and b_j,

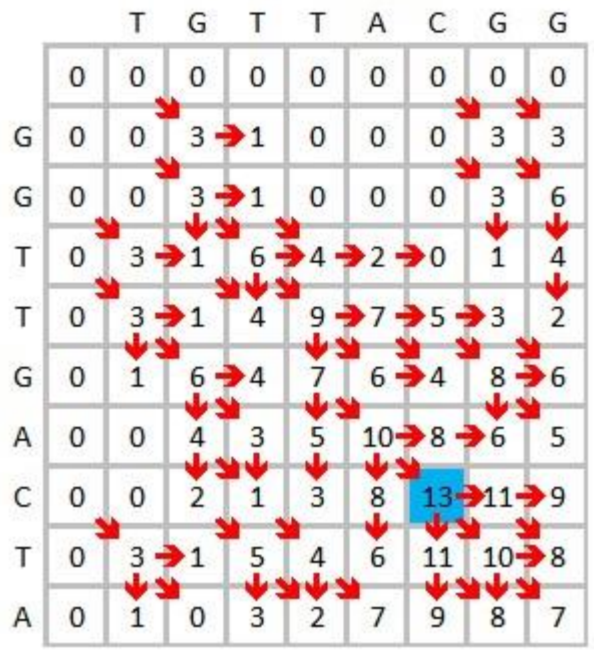
$H_{i-k,j} - W_k$ is the score if a_i is at the end of a gap of length k,

$H_{i,j-l} - W_l$ is the score if b_j is at the end of a gap of length l,

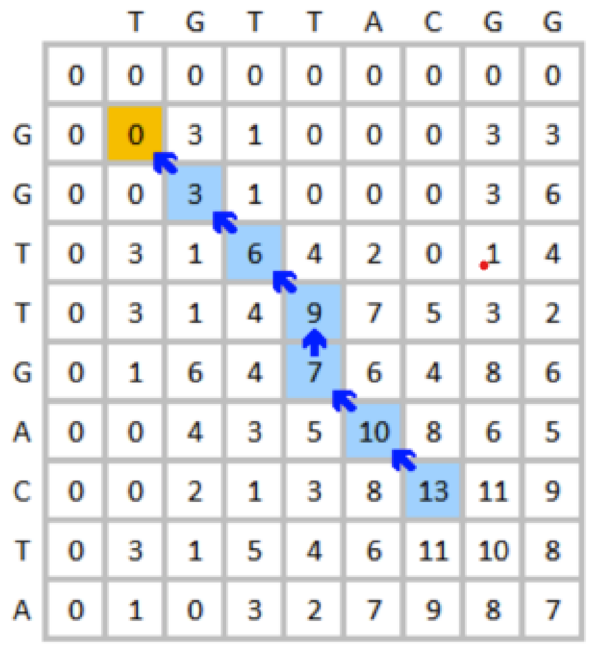
0 means there is no similarity up to a_i and b_j.







		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7



		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

The Alignment result is:

G T T – A C
G T T G A C

Need for Parallel

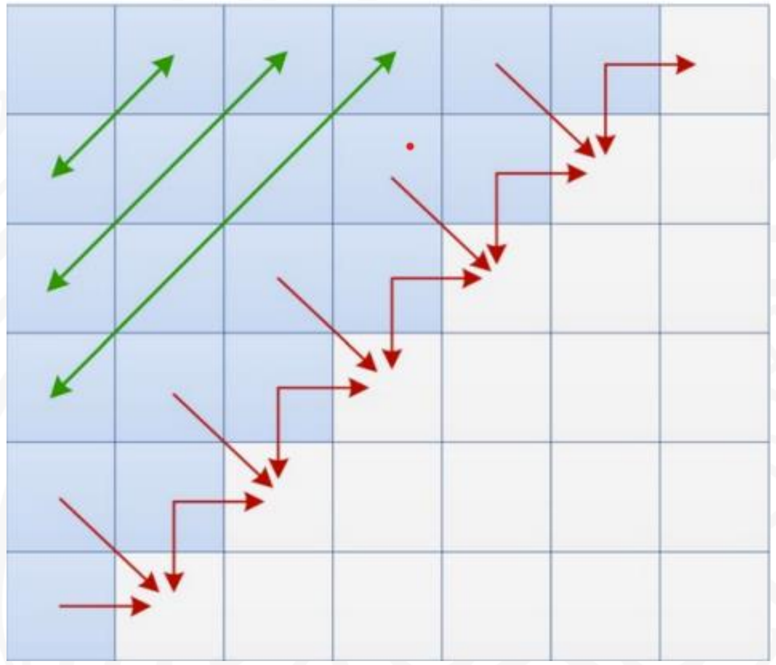
- Sequence alignment is a highly time-consuming task as it takes quadratic time which is O(mn), m,n are the lengths of the protein sequences
- Let’s take an example of (COVID-19), scientists identified its common features by aligning it against other viruses

University at Buffalo The State University of New York

Parallel Implementation

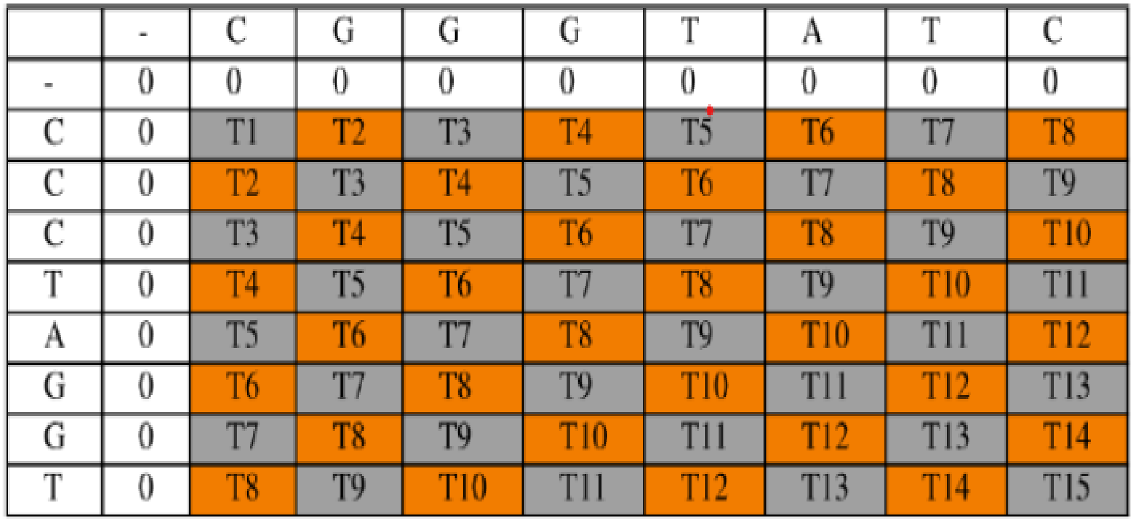
Wavefront Approach:

- Each cell in the computation matrix depends on the calculations of three other cells, as indicated by red arrows in the document.
- Diagonal elements doesn't depend on the other diagonal elements so they can be computed parallelly



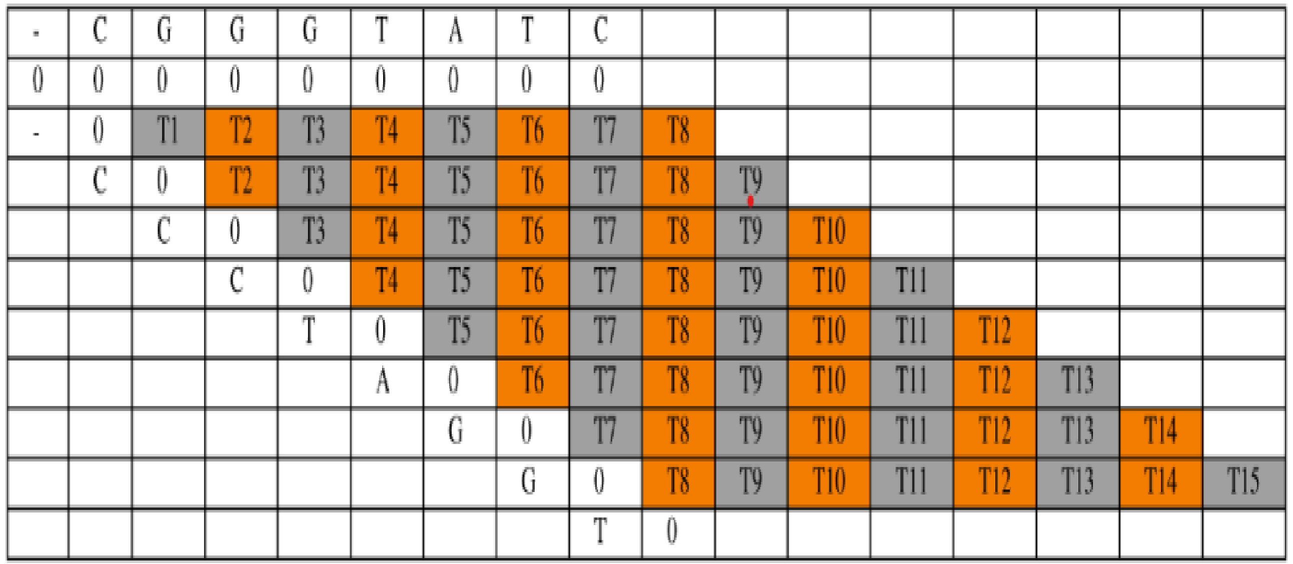
Does this work for OpenMP?

- Yes, it works but with doing some minor modifications like converting the whole matrices into blocks and executing them parallelly which is called as Tile Parallelization but the main drawback with this approach is load imbalance



	-	C	G	G	G	T	A	T	C
-	0	0	0	0	0	0	0	0	0
C	0	T1	T2	T3	T4	T5	T6	T7	T8
C	0	T2	T3	T4	T5	T6	T7	T8	T9
C	0	T3	T4	T5	T6	T7	T8	T9	T10
T	0	T4	T5	T6	T7	T8	T9	T10	T11
A	0	T5	T6	T7	T8	T9	T10	T11	T12
G	0	T6	T7	T8	T9	T10	T11	T12	T13
G	0	T7	T8	T9	T10	T11	T12	T13	T14
T	0	T8	T9	T10	T11	T12	T13	T14	T15

Figure 2. Cases calculable at the same time T_i.



-	C	G	G	G	T	A	T	C								
0	0	0	0	0	0	0	0	0								
-	0	T1	T2	T3	T4	T5	T6	T7	T8							
	C	0	T2	T3	T4	T5	T6	T7	T8	T9						
		C	0	T3	T4	T5	T6	T7	T8	T9	T10					
			C	0	T4	T5	T6	T7	T8	T9	T10	T11				
				T	0	T5	T6	T7	T8	T9	T10	T11	T12			
					A	0	T6	T7	T8	T9	T10	T11	T12	T13		
						G	0	T7	T8	T9	T10	T11	T12	T13	T14	
							G	0	T8	T9	T10	T11	T12	T13	T14	T15
								T	0							

Figure 3. Linear representation of the parallelizable boxes.

Approach for OpenMP

Row or Column Parallelism:

- Given the shared memory environment, this simpler strategy might now be more viable, especially if combined with OpenMP's ability to dynamically assign rows or columns to threads based on current workload, thus addressing the load balancing issue.
- In row parallelism, the task of filling this matrix is divided among several threads, with each thread responsible for computing one or more entire rows of the matrix.
- While threads can compute the scores within their assigned rows in parallel, the computation of each row has to wait until the previous row is fully calculated.

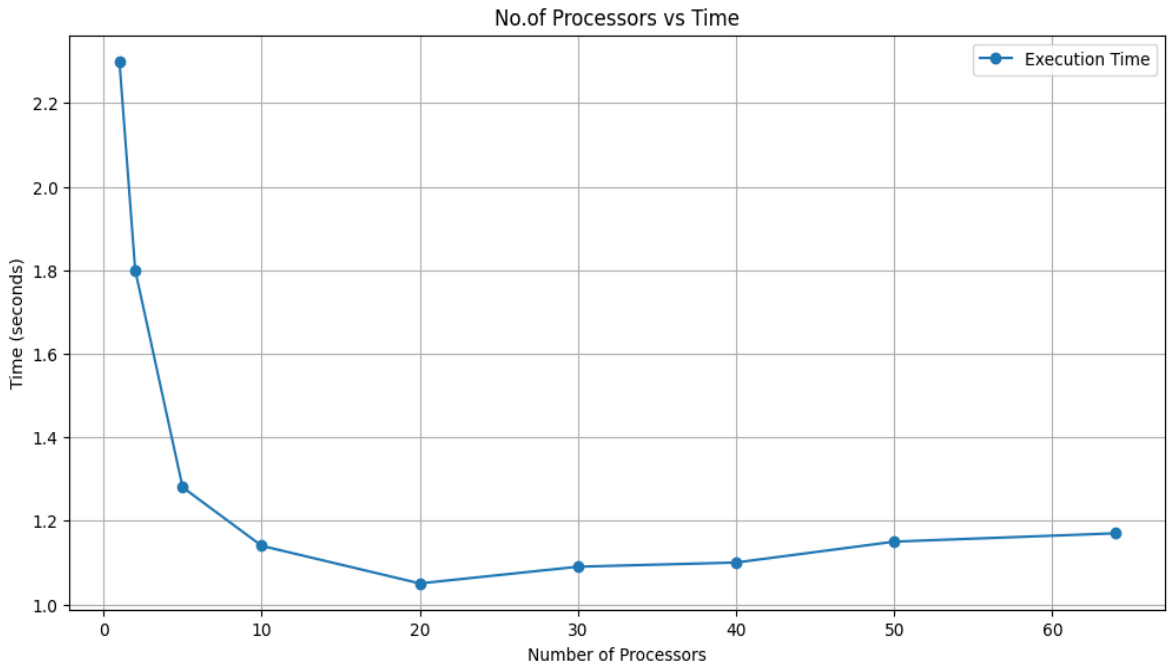
Slurm Script:

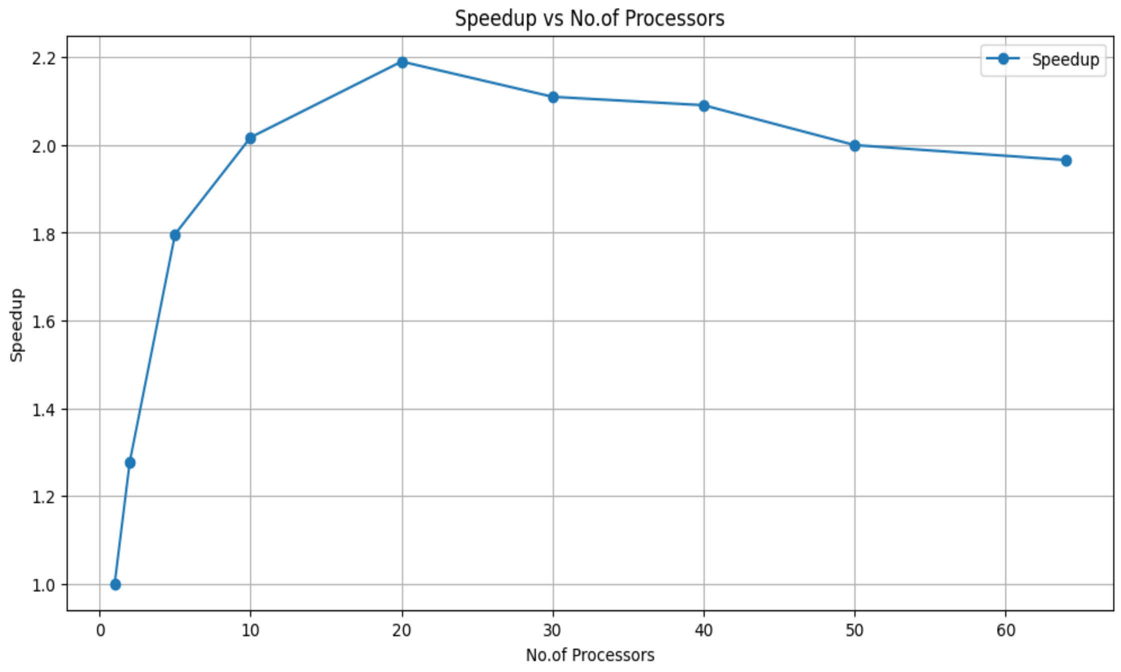
```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --time=00:10:00
#SBATCH --partition=general-compute
#SBATCH --qos=general-compute
#SBATCH --cluster=ub-hpc
#SBATCH --reservation=ubhpc-future
#SBATCH --job-name="sma_4_threads"
#SBATCH --output=output_openmp.txt
#SBATCH --exclusive
module load intel
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

gcc -fopenmp -o compiled_file sma_openmp.c
./compiled_file
```

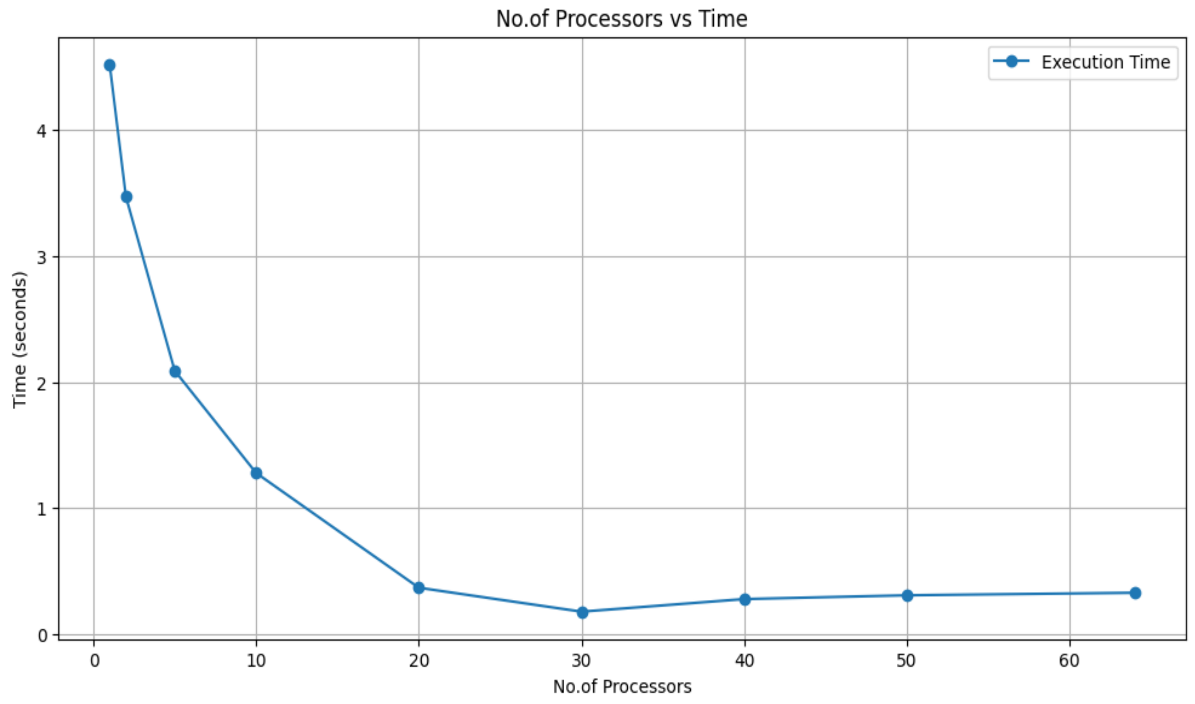
Results

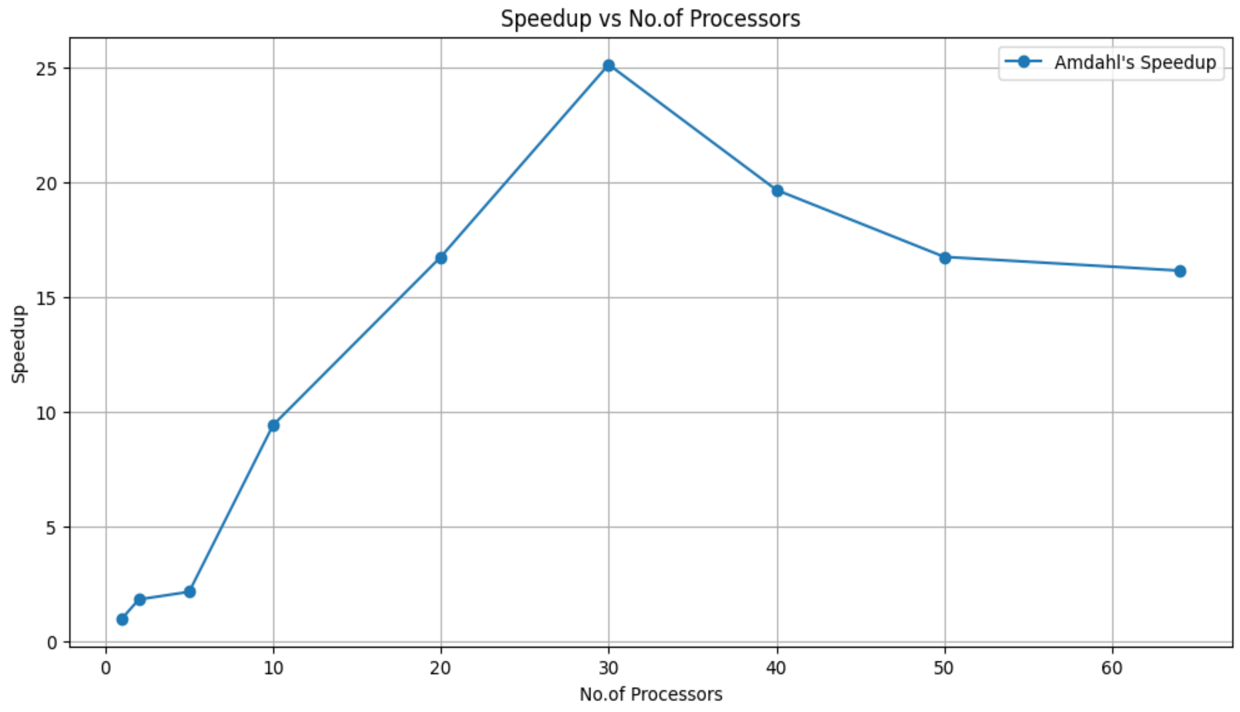
For dataset N = 1000



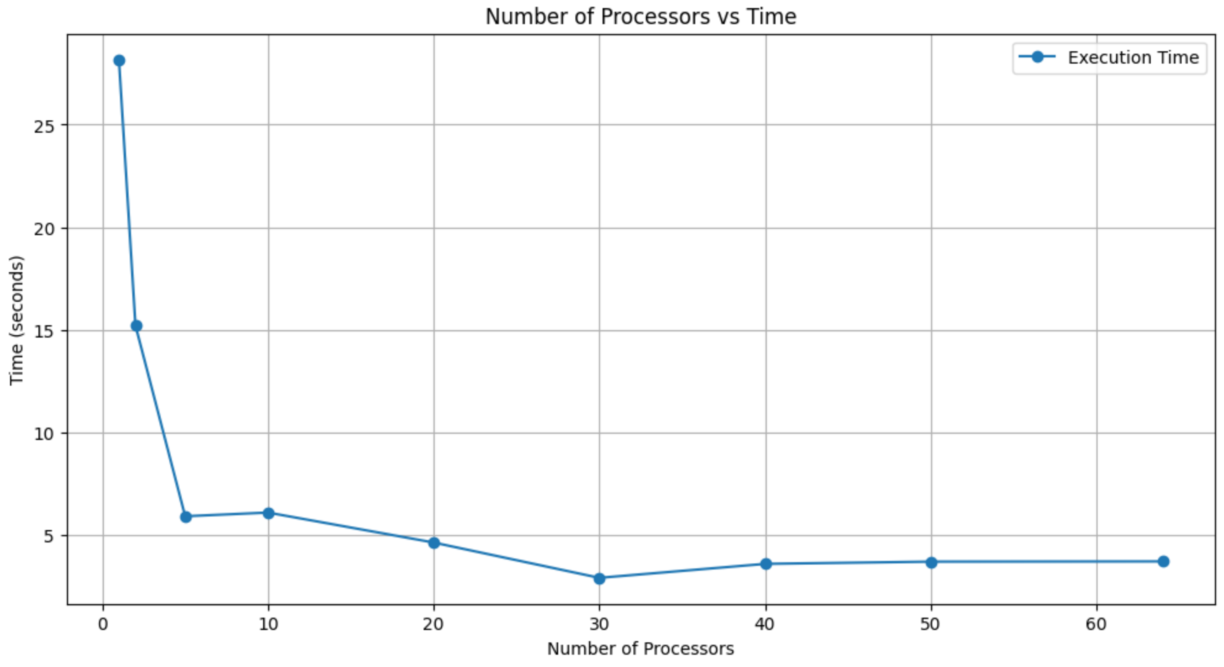


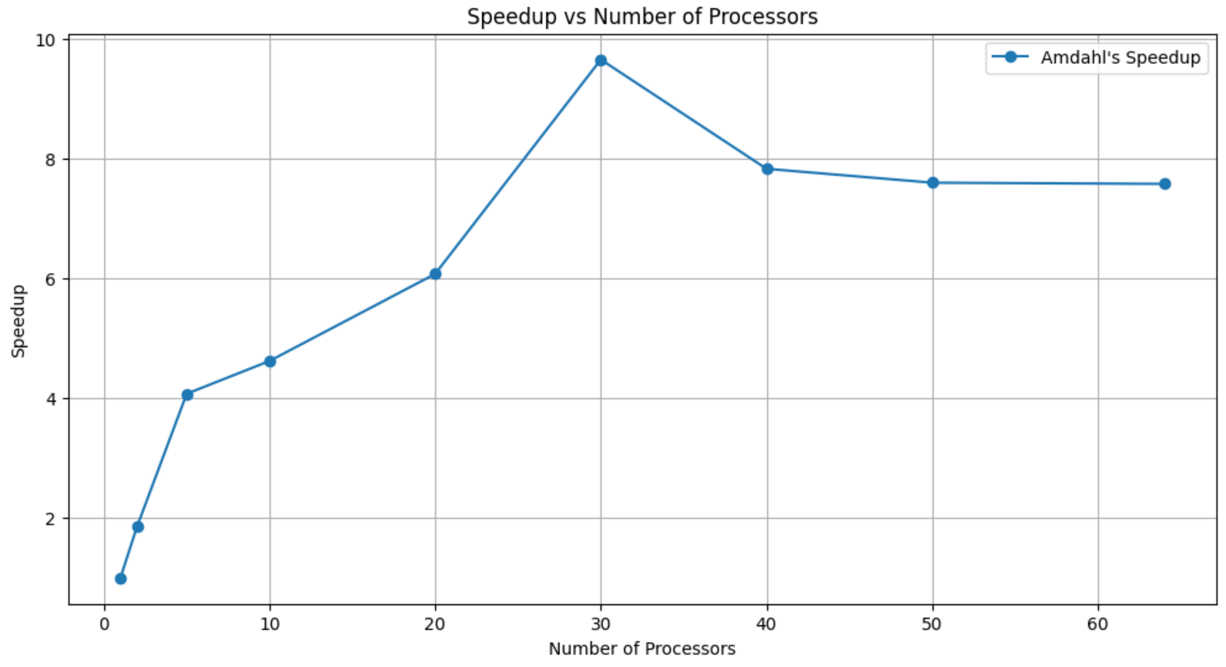
N = 2000


No.of Processors vs Time
Execution Time
Time (seconds)
No.of Processors
0
1
2
3
4
0
10
20
30
40
50
60

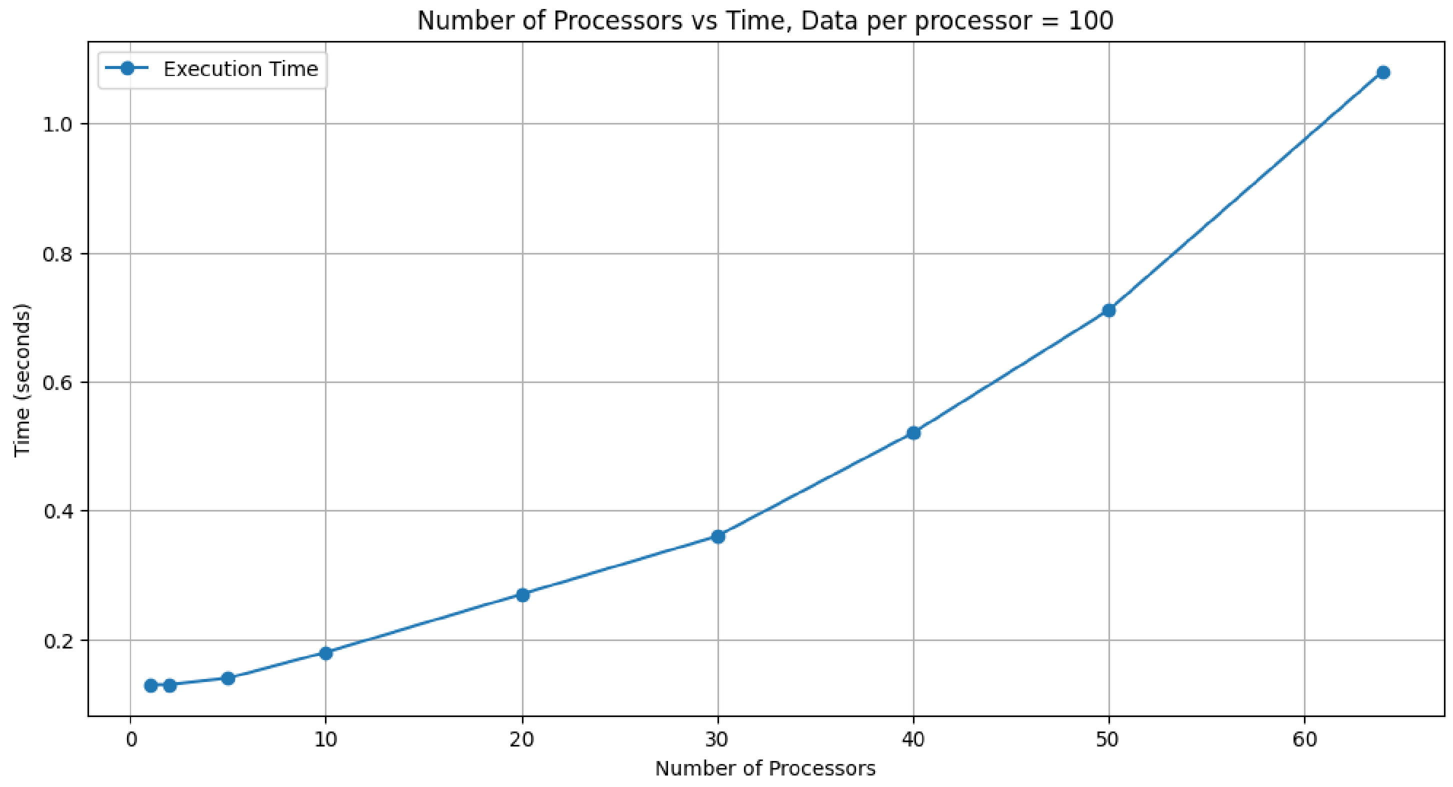

Speedup vs No.of Processors
Amdahl's Speedup
Speedup
No.of Processors
0
5
10
15
20
25
0
10
20
30
40
50
60

N = 4000


Number of Processors vs Time
Execution Time
Time (seconds)
Number of Processors
0
10
20
30
40
50
60
5
10
15
20
25


Speedup vs Number of Processors
Amdahl's Speedup
Speedup
Number of Processors
0
10
20
30
40
50
60
2
4
6
8
10

Gustafson's Law:



References

- Chaibou, Amadou, and Oumarou Sie. "Comparative study of the parallelization of the Smith-Waterman algorithm on OpenMP and Cuda C." Journal of Computer and Communications 3.06 (2015): 107.
- https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm
- Parallelizing the Smith-Waterman Algorithm using OpenSHMEM and MPI-3 One-Sided Interfaces - Matthew Baker, Aaron Welch, Manjunath Gorentla Venkata.

Thank You