SMITH-WATERMAN ALGORITHM OPENMP

Sai Ram Gurram
CSE 633 (Dr. Russ Miller)
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 $O(mn)$, where m and n are the lengths of the protein sequences.
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

\[
H_{ij} = \max \begin{cases} 
H_{i-1,j-1} + s(a_i, b_j), \\
\max_{k \geq 1} \{H_{i-k,j} - W_k\}, \\
\max_{l \geq 1} \{H_{i,j-l} - W_l\}, \\
0 
\end{cases} \quad (1 \leq i \leq n, 1 \leq j \leq m)
\]

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 \(b_j\).
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 $O(mn)$, where $m,n$ 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.
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 don’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.

Figure 2. Cases calculable at the same time T_j.

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:

```bash
#!/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 $N = 1000$
$N = 2000$
$N = 4000$
Gustafson's Law:

Number of Processors vs Time, Data per processor = 100
References


• 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