Explanation

**By: Eylon Mizrahi**

I have implemented the solution by integrating MPI OMP and CUDA to work together.

I decided to divide the work of each process by offsets of the sequence via MPI, this means that for example: for 2 processes and 1000 offsets, each will get to work on 500 offsets.

All offsets will be calculated for each mutation and they will be sent to the CUDA device in order to create a string of Signs for each combination of offset and mutation.

(A.k.a the Signs \* : . '' from the exercise)

This string of signs will then be copied back to the host, and OMP threads will then compute the final alignment score on it. This process of creating a combination, sending it to CUDA to create a string, and receiving it back in order to compute the score will happen for each and every combination and for every sequence in the input.

This is all encapsulated within a loop that will determine the best stats for each sequence – best score best offset and best mutation.

Every Process has an answer to only half of the combinations, so process ID > 0 will send the best results to process with ID 0.

The Process with ID 0 will then decide the absolute best statistics for each sequence in the input.txt.

Structs in my project:

MainSequence:

holds the First Sequence its length and the 4 weights for the algorithm.

Sequence:

holds a single small sequence and its length.

IterationInfo:

holds the rank of the holding MPI process, the number of MPI processes in the system, current offset, and current mutation for this iteration.

BestStats:

holds the best information the current MPI process holds, aka best score best offset and best mutation.

Cuda Note:

The computation in cuda is as follows:

each GPU thread will calculate a similarity character in a given position in the target signs array, resulting in a full Signs array after each thread is executed.

OMP Note:

The computation in OMP is as follows:

each OMP thread will calculate his fair share of similarity characters in the ready signs array, and when they all join, they sum up the score together.

Complexity:

Given only one Main Sequence of length M and one small Sequence of length N.

The complexity of the naive algorithm is:

Given K MPI processes split between the (M - N + 1) offsets, X OMP threads split between Signs array of size N each time, and Y cuda threads that fill Signs array of size N concurrently.

The complexity of my solution is:

Note: the basic sequential algorithm took 91.7 seconds

while the basic parallel solution took 37.92 seconds and the best solution on 2 computers took 26.69 seconds.

