GPU-REMuSiC: the implementation of Constrain Multiple Sequence Alignment on Graphics Processing Units

Yu-Shiang Lin[1] , Chun-Yuan Lin[1,*], Sheng-Ta Li[1], Jon-Yu Lee[3] and Chuan Yi Tang[2]
[1]Department of CSIE, Chang Gung University, Taoyuan Taiwan
[2]Department of CS, National Tsing Hua University, Hsinchu Taiwan
[3]NVIDIA corporation, Taipei Taiwan [*]Corresponding author: cyulin@mail.cgu.edu.tw

Introduction

Sequence alignment is a fundamental and important research filed in the computational biology. Dynamic programming (DP) algorithms, such as Needleman-Wunsch (NW), have been proposed to align two biology sequences. **RE-MuSiC** is a constrained multiple sequence alignment (CMSA) method based on DP results; it has been proposed to use regular expression constraints to find the important function sites. We have implemented RE-MuSiC tool on multi-GPUs (called **GPU-REMuSiC**) with NVIDIA CUDA and concluded eight computational models for a DP computation by intra-task parallelization on GPUs.

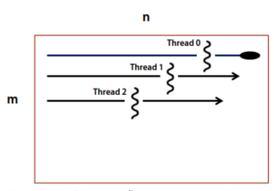
Needleman-Wunsch algorithm

If we use $\boldsymbol{A(i,j)}$ to represent an **optimal alignment** of $S_1[1,i]$ and $S_2[1,j]$, it can be represented as a recursive function equation such that

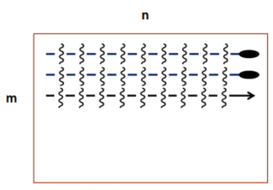
$$\boldsymbol{A(i,j)} = \mathbf{max}\begin{cases} \boldsymbol{A(i-1,j-1)+\sigma(i,j)} \\ \boldsymbol{A(i-1,j)+\sigma(i,-)} \\ \boldsymbol{A(i,j-1)+\sigma(-,j)} \end{cases}$$

Computational models for intra-tasks

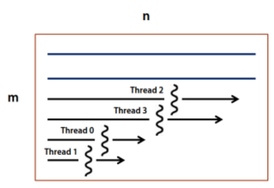
According the characteristics of GPU, a DP computation by **intra-task parallelization** can be computed with one of eight models: *SRST*, *SRMT*, *SDST*, *SDMT*, *ARST*, *ARMT*, *ADST*, *ADMT*. These models are not formally defined in the past and we use the ***SRMT*** model to implement GPU-REMuSiC.



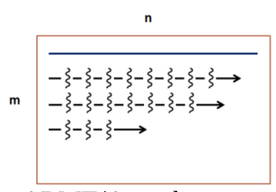
SRST(Synchronous Row Single Thread)



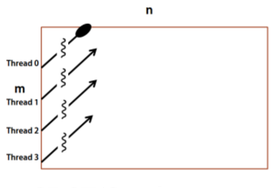
SRMT(Synchronous Row Multiple Threads)



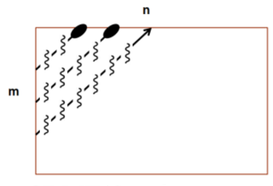
ARST(Asynchronous Row Single Thread)



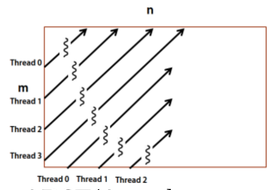
ARMT(Asynchronous Row Multiple Threads)



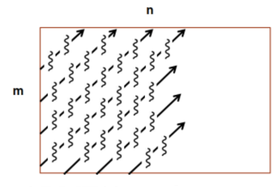
SDST(Synchronous Diagonal Single Thread)



SDMT(Synchronous Diagonal Multiple Threads)



ADST(Asynchronous Diagonal Single Thread)

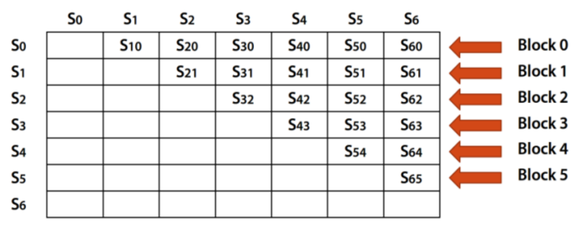


ADMT(Asynchronous Diagonal Multiple Threads)

Distance matrix

In the distance matrix, **thread blocks** can simultaneously execute one or more working cells, i.e. a sequence alignment computation by NW algorithm. For the thread blocks and working cells, we proposed an assignment method as follows.

Assume that there are seven sequences $S_0 \sim S_6$ to construct a **distance matrix** by NW algorithm. This matrix can be formed as a triangular block. The distribution of a thread block is responsible for the working cells of a row. After finishing the working cells of a row, a thread block will be assigned with a next row or be terminated to release GPU computing resources.

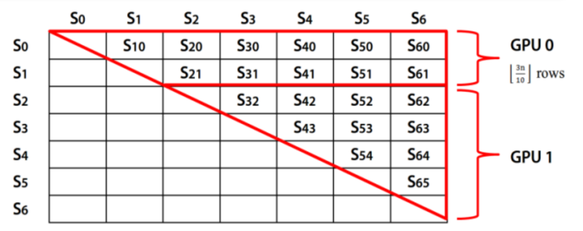


Scoring Matrix

There are three commonly used **scoring matrices** for the protein alignment, GONNET, BLOSUM and PAM. We pre-query any two characters (among all residues) from a scoring matrix and store the query results in score_table array, constructed as 26 × 26 elements. We use the hash function proposed by Striemer and Akoglu (2009) to calculate the alignment score for c, d characters of S_i and S_j. ($\sigma(i,j) = \mathrm{SCORE}(\mathrm{ascii(c)} - 65, \mathrm{ascii(d)} - 65)$). Since the scoring table will be read frequently, it must be pre-allocated in the **constant memory**.

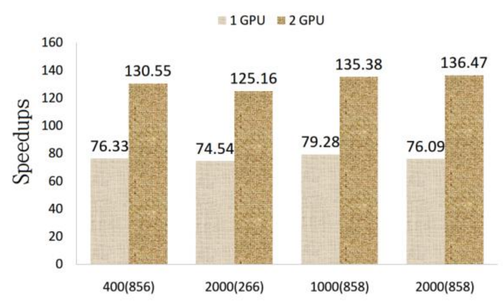
Load Balancing Multi-GPUs

The distance matrix will be **divided and computed** by two graphics cards. If there have n sequences, the cut edge is set on the $\left\lfloor \frac{3n}{10} \right\rfloor$ rows which divided the matrix into two blocks. $\frac{3}{10}$ is the approximate factor.

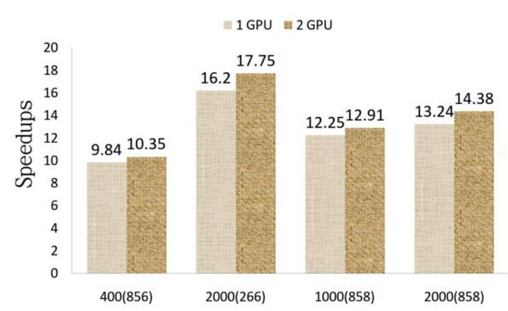


Speedups

We implemented the GPU-REMuSiC on two NVIDIA GeForce GTX 260 and installed in a PC with an Intel i7 920 CPU and 12GB RAM running the Linux OS. The test data are randomly generated protein sequences, such as 400 sequences in 856 characters length. We set up 512 and 192 for the number of thread blocks and per block threads, respectively.



(a) DP computing time



(b) overall computing time