Project Proposal: Implementation of Smith-Waterman Algorithm in SIMT using CUDA

Abstract: This proposal aims to improve the computational efficiency of the Smith-Waterman algorithm for local protein sequence alignment by taking advantage of CUDA's parallel computing capabilities. The algorithm will be implemented in both sequential C and parallelized CUDA versions, focusing on optimizing the scoring phase utilizing the BLOSUM62 substitution matrix and affine gap penalties. The performance of both versions will be assessed by comparing execution times and validating the CUDA implementation. The goal is to show that GPU acceleration may significantly reduce execution time, demonstrating its potential for upgrading computationally complex bioinformatics algorithms.

Project Description

Sequence alignment is a technique used to compare nucleic acid or protein sequences in order to identify regions of similarity between the sequences. The Smith-Waterman Algorithm is one such algorithm for sequence alignment, specifically local sequence alignment. Instead of comparing entire sequences, the algorithm looks for and compares segments of all possible lengths to identify similarities and obtain a measure of similarity (Smith et al., 1981). However, sequence alignment is computationally costly since its computing and memory requirements increase significantly as the database or queries increase, resulting in high execution time.

Researchers have tried a variety of methods to speed up the Smith-Waterman algorithm. One study employs reconfigurable computing architectures based on FPGA devices. This implementation has demonstrated performance gains of up to 28 times above standard microprocessor-based methods (Storaasli et al., n.d.). Another study proposed a SIMD implementation, which achieved estimated execution speeds of up to 23.8 GCUPS, rivaling FPGA implementations (Rudnicki et al., 2008).

With the advancement of hardware technology, GPUs provide new possibilities for improving algorithms. CUDA's parallel computing capabilities can improve the Smith-Waterman algorithm's computing efficiency. The proposed project aims to utilize the SIMT paradigm using CUDA to implement and parallelize the algorithm. Protein sequences will be the choice of input data to be compared and aligned instead of nucleic acid sequences (DNA or RNA). With that, the group will also utilize the BLOSUM substitution matrix instead of the simple substitution

matrix to score the similarities of the protein sequences better. Lastly, since the linear gap penalty applies the same penalty to a single large gap as to multiple minor gaps, the group will utilize Affine gap penalties to designate insertion/deletion scores instead of a linear gap penalty. Unlike linear gap penalties, the affine gap penalties are gap length-dependent (Rotcha et al., 2018).

The group will implement and execute the algorithm in two separate kernels, one sequentially in C as a baseline and the other utilizing CUDA, accounting for the required changes to make efficient use of the parallelization present in CUDA. The respective latencies of both kernels are then recorded several times in order to verify the computing efficiency of both algorithms properly. The results will then be analyzed to validate whether or not the CUDA implementation will be more efficient than the C implementation, as well as to score the correctness of the algorithm itself.

Proposed Algorithm

The Smith-Waterman Algorithm is divided into 4 steps. First is determining the substitution matrix and gap penalty scheme to be used. As previously mentioned, the choices for this project are the BLOSUM 62 substitution matrix, as well as the affine gap penalty scheme. Second is the initialization of the scoring matrix: said matrix is of size length_S1 + 1 by length_S2 + 1, since the first row and column are initialized to 0. This allows any sequence to be freely aligned to any other. Third is the scoring, and is particularly the part of the algorithm that will be parallelized. The recurrence relation for this algorithm is shown below.

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

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.

In general, the value of a matrix element is taken from the substitution matrix; ergo, the compared ith and jth protein sequence values will have a corresponding value from the BLOSUM 62 matrix. This is added to the score from the (i-1)th and (j-1)th index to account for the previous element's score. Alternative scores are also computed for when a gap is introduced in either sequence, wherein only the non-gapped dimension is incremented, and the gap penalty is introduced, calculated using the selected affine gap penalty scheme. The max function is called across the 3 results since differing best-case scores can be obtained depending on which gap assumption is used. 0 is also included to ensure no negative scores. Lastly, a traceback is performed, starting at the matrix element with the best score, going back to the source of each score recursively until 0. This provides the sequence segment with the best local alignment. Succeeding best local alignments can be obtained by starting the traceback at the matrix element with the second highest score outside the best alignment's trace.

References

Rocha, M., & Ferreira, P. G. (2018). Pairwise Sequence Alignment. Elsevier EBooks, 133–162. https://doi.org/10.1016/b978-0-12-812520-5.00006-7

Rudnicki, W. R., Jankowski, A., Modzelewski, A., Piotrowski, A., & Zadrożny, A. (2009). The new SIMD Implementation of the Smith-Waterman Algorithm on Cell Microprocessor. Fundamenta Informaticae, 96(1-2), 181–194. https://doi.org/10.3233/fi-2009-173

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. Journal of Molecular Biology/Journal of Molecular Biology, 147(1), 195–197. https://doi.org/10.1016/0022-2836(81)90087-5

Storaasli, O., Strenski, D., & Inc, C. (n.d.). `Accelerating Genome Sequencing 100X with FPGAs Figure 2. Virtex-II Pro 50 FPGA speedup Figure 3. Virtex-4 LX160 speedup Cray XD1 (Virtex2) Speedup.